Graduation Internship Report

Engineering college SUP'Galilée

Paris Nord XIII University

Alternative Energies and Atomic Energy Commission

CEA Saclay - DEN/DANS/DM2S/STMF/LMEC

From 1st April 2015 to 30^{th} September 2015

Discipline: Applied Mathematics

Option: scientific calculations

Environment of Simulation for Boiling Flows in Differentially Heated Parallel Channels

Prepared by : **Tassadit SID ABDELKADER**

Responsibles:

CEA Tutor : Michaël NDJINGA, Research Engineer CEA

University Tutor : Olivier LAFITTE, University Professor, Paris Nord XII

Viva's date: 25^{th} of September 2015

Abstract

For the sake of getting an engineering degree in applied mathematics and scientific computing in Sup'Galilée College, Paris Nord University, I conducted a graduation internship entitled: *Environment of Simulation for Boiling Flows in Differentially Heated Parallel Channels*. In Cea-Saclay, from 1st April 2015 to 30th September 2015.

I already conducted a graduation project (from December 2014 to February 2015) of research and literature review (bibliography) on the prototyping of industrial code ***FLICA 4*** in 1D, in order to improve the robustness. One of the conclusions of this work is that a specific treatment is crucial to the stiff source terms for specific thermal hydraulics cores of nuclear reactors (well balanced schemes).
The current graduation internship aims at confirming these conclusions in case of 2D, 3D particularly in the configurations presenting a recirculation on the parallel channels. We also hope to compare the well balanced schemes proposed by the graduation internship with those precise Low Mach number schemes.

In this internship, I started by debugging, correcting and validating the developed model in the PFE, which enabled me to gain in competence in Petsc library and the Eclipse IDE. The second component of the internship was to develop the CoreFlows application to handle any geometry (1D , 2D and 3D) , and impose boundaries conditions. For this, I had to to take in the hand several open sources libraries, mesh handling, scientific computing and visualization (Salomé, CDMATH , PETSc) and software development tools . And finally, after a milestone study and implementation, I could validate the correct operation of CoreFlows and compare the proposed numerical methods (well balanced schemes and Low-Mach schemes).

The present report is composed of five main sections, the first one describes the internship's context and the host structure(CEA). The second includes mathematical modeling of multiphase flows concepts, which is briefly reviewed with a focus on balance equations. The third section contains some numerical methods and schemes that we have studied. The fourth describes the CoreFlows application with a detailed description of all the software engineering tools WHICH have been used. The fifth section presents the findings of the internship project. To conclude, some conclusions and future work recommendations are suggested.

Key Word : Two-phase flow - Fluid Mechanics - Drift model - Numerical Simulation - Roe Schemes - Well-Balanced schemes - Low-Mach scheme - Staggered Mesh - Open Source software - Object-oriented programming

Acknowledgement

First of all, special thanks goes to my tutor, Micheal NDJINGA, who, by his competences, experiences and his nobel human characteristics, guided me throughout this humble work. I also want to show my gratitude to Professor Olivier LAFITTE for being there by my side while conducting this internship.

And then I wish to thank the set of MACS and SUP'Galilée professors and particularly, the two responsibles, Olivier LAFITTE and Emanuel AUDUSSE, for their encouragement and the knowledge they provided for me.

Thanks to all LMEC laboratory' members for their help, guidance, advice and encouragement, and precisely, Anouar MEKKAS. Also I want to show my gratitude to Pascal OMNES, Adrien BRUNETON, Marc TAJCHMAN and Francis KLASS. In addition to my colleagues trainees for the experiences and knowledge we shared during this training.

Finally, I would like to thank my family members who were always there by my side, though the distance between us they were there to advise, support, encourage and motivate me.

Nomenclature

We denote by the index:

- $k = l$ the physical quantities related to the liquid phase,
- $k = v$ the physical quantities related to the vapor phase.

The physical quantities related to the thermodynamic

Symbol	Signification	Unity (SI)
ρ_k	density	$kg.m^{-3}$
α_k	void fraction	1
e_k	internal energy	$J.kg^{-1}$
$h_k = e_k + P/\rho_k$	specific enthalpy	$J.kg^{-1}$
$H_k = h_k + \lvert u_k \rvert^2/2$	specific total enthalpy	$J.kg^{-1}$
$\bar{\bar{\sigma}}_k$	viscosity tensor	$kg.m^{-1}.s^{-2}$
$h_{k,sat}$	specific enthalpy at saturation	$J.kg^{-1}$

The physical quantities related to the mixture

Symbol	Signification	Unity (SI)
$\rho = \sum \alpha_k \rho_k$	density	$kg.m^{-3}$
$\vec{U} = \frac{\sum \alpha_k \rho_k \vec{V_k}}{\rho}$	velocity vector	$m.s^{-1}$
$\vec{U_r} = \vec{u_v} - \vec{u_l}$	relative velocity	$m.s^{-1}$
$C_v = \frac{\alpha_v \rho_v}{\rho}$	vapor mass concentration	1
$e = \frac{\sum \alpha_k \rho_k e_k}{\rho}$	internal energy	$J.kg^{-1}$
$E = C_v E_v + (1 - C_v) E_l$	total energy	$J.kg^{-1}$
$h = \frac{\sum \alpha_k \rho_k h_k}{\rho}$	specific enthalpy	$J.kg^{-1}$
P	pressure	Pa

Sommaire

Chapter 1

General introduction

1.1 Internship Context

1.1.1 CEA

The CEA is the French Alternative Energies and Atomic Energy Commission (Commissariat à l'énergie atomique et aux énergies alternatives), which is the French scientific research institution specialized in four wide fields: Low-carbon energies, Defense and Global energy, Information technologies and Health technologies.
It is composed of five main divisions:

- Nuclear Energy Division (NED)
- Military Applications Division (MAD)
- Technological Research Division (TRD)
- Life Sciences Division (LSD)
- Material Sciences Division (MSD)

It was created in October 18^{th}, 1945 by General De Gaulle, naming the high commissioner in atomic energy Frédéric Joliot-Curie as a president. Today, it includes more than 15.000 employees in different centers situated allover France. (Figure 1.1 illustrates the location of CEA's centers throughout France). Divided into two types of centers, one is for the Military Applications' studies center whereas the other one is for the Civilian' Studies center, CEA- Saclay where I had my internship is a part of this latter. In addition to this there is a center which includes the biggest number of staff.

CEA has various partnerships with research organizations, local collectivities and universities. In this regard, it is an integral part in national alliances coordinating the French research in the field of energy (ANCRE), Life and Health sciences (AVIESAN), Numerical sciences and technologies (ALLISTENE) and environment sciences (AllEnvi).

As a leader in applied research, CEA hosts (center of Bruyeres le Chatel) one of the most powerful supercomputers in Europe. With its 11,520 processors, 92160 cores of its

calculations and its gigantic memory of 360 terabytes, the CURIE supercomputer can reach speeds of 2 Petaflop[1]. CURIE is part of this network GENCI[2].

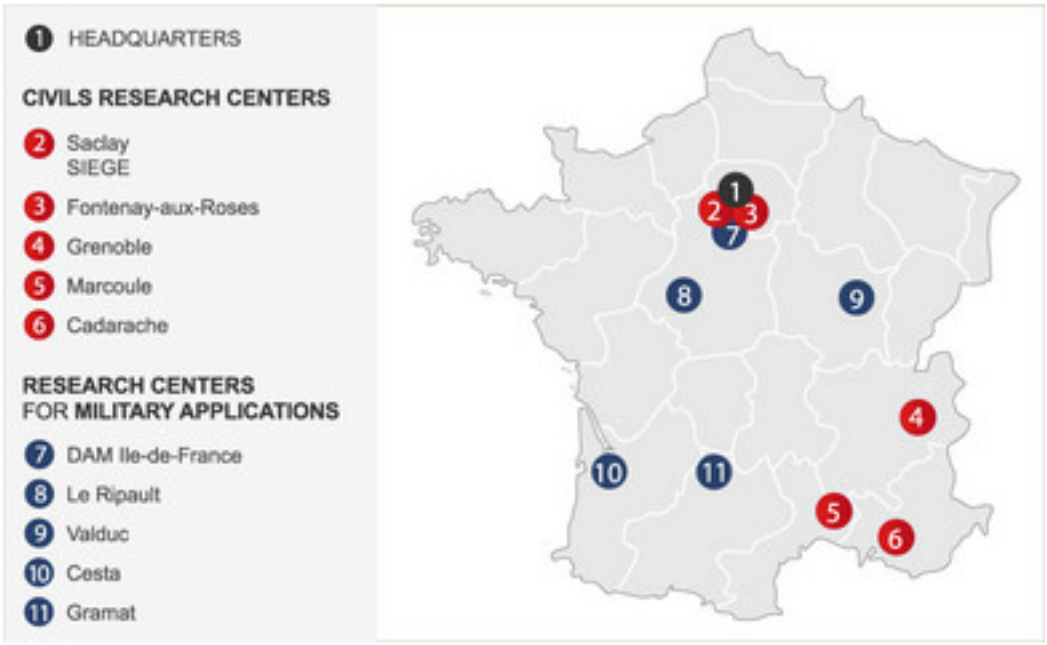

Figure 1.1: The ten CEA centers in France

CEA also hosts the INSTN whose mission is teaching and professional internship in the nuclear field, as well as many organizations or companies:

1.1.2 Saclay Center

CEA-Saclay Center is one of the most important research centers and development in Europe. It is situated in the Essonne department (ile de France). The center is 220 ha wide, located in the Saclay plateau in the scientific and technological pole of Paris-Saclay.

It consists of a main area which hosts CEA's headquarters and an extension "l'Orme de Merisiers".

It was inaugurated in 1952 in order to activate the first nuclear research reactor, successor of Zoe battery : EL2[3].

It contains approximately 400 buildings, nearly 8000 people who run daily there within it, 5400 CEA and others coming from different institutions and companies.

CEA also hosts the INSTN[4] whose mission is teaching and professional internship in the nuclear field, as well as many organizations or companies:

1. 1 Petaflop = 10^{15} flops (i.e FLoating point Operation Per Second)
2. GENCI : Grand Équipement National de Calcul Intensif
3. Heavy water reactor n°2
4. Institut national des sciences et techniques nucléaires

- Agency for Radioactive Waste Management(Andra).
- International Institute of Nuclear Energy(I2EN).
- Iba, Producer of Radiolabel Molecules for Radiotherapy.
- Radiation Protection and Nuclear Safety (IRSN).
- Areva. TA (Formerly Technicatome).
- Euriso-Top, Solvet Containing Deuterium and Banded products Producer.

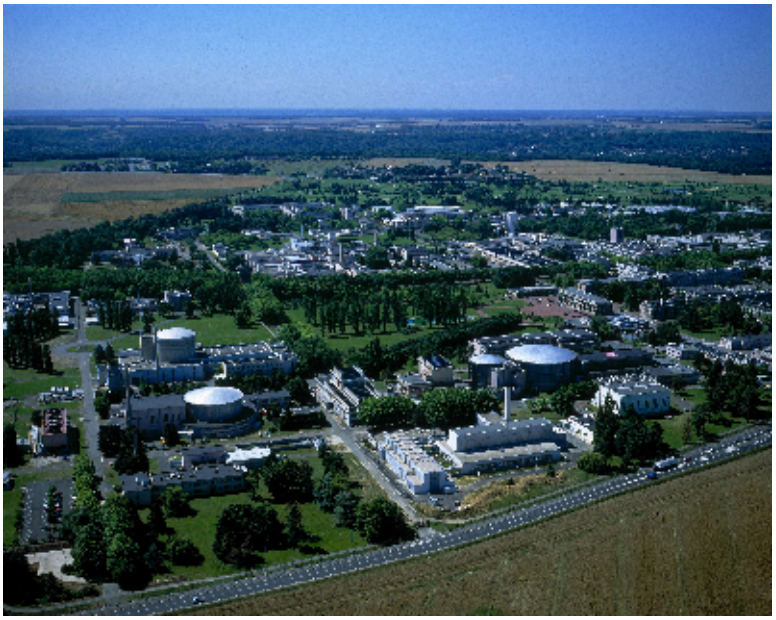

Figure 1.2: Cea-Saclay center, view of the sky

CEA-Saclay operates in different domains from which we can list: Nuclear energy,Life sciences, Material sciences, Climate and Environment, Technological Research and Education. In addition to these there are other research activities in Hydrogen and Bio-energies. Furthermore, other works can be cited such as: researches on radioactive waste management and the demolition of old nuclear research reactors.

In addition to the field of energy studies, other research areas are studied in CEA-Saclay:

- Technological research: embedded computing systems, interactive systems (man-machine relationships), sensors and signal processing.
- Health researches: the effect of radiation on cells and molecules, protein engineering, medical imaging research and radioimmunoassay.
- Environmental studies: climate modeling, and the Global warming effects.

1.1.3 Modeling Laboratory on a Components Scale

The internship has been conducted in LMEC laboratory (Laboratoire de Modèlisation à l'Échelle Composante) which is part of "service of Thermal-hydraulics and Fluid Mechanics "(STMF) which includes nearly 130 employees. STMF is part of the "Systems and Structures Modeling Department" which includes 300 employees which is part of " Delegated Management to Saclay Nuclear Activities" within the "Nuclear Energy Division" (DEN), which includes 5000 employees.

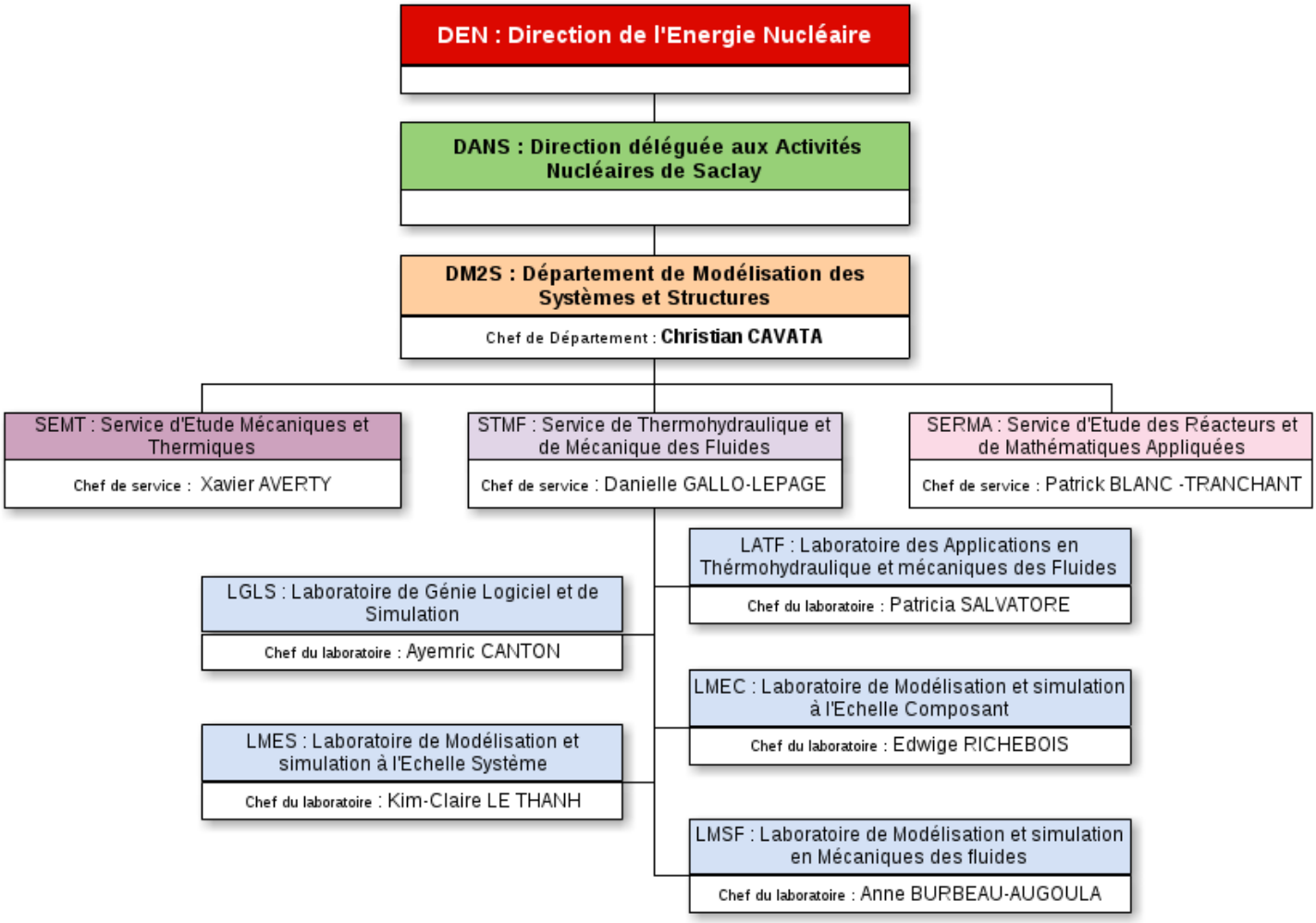

Figure 1.3: CEA's organizational chart

The twenty members of the laboratory organized into teams work on several aspects: Modeling, development, simulation and maintenance of industrial codes.

Competencies are situated essentially in the following fields:

- Physical modeling of two-phase flows,
- Numerical analysis,
- Design support for the safety analysis of reactors,

- Software engineering.

1.2 Problem Statement

Among CEA activities, technologies related to the production of nuclear energies are a crucial part of CEA' original mission.

Working principle of a pressurized water reactor PWR France has 19 nuclear stations, in which, energy is produced by "Pressurized Water Reactors " and the term " pressurized water" is coming from the fact that water which carries the heat in the primary circuit (see figure 1.4) is under a very strong pressure: 155 bars which means 155 times the atmospheric pressure

Therefore, their principle operation is as follows:

1. A nuclear reaction occurs in the reactor's core and has a very significant heat,
2. The heat generated is transferred to water of primary circuit (figure 1.4). This water is about 300°C.
3. Water of primary circuit in turn transfers its heat to the one of the secondary circuit (figure 1.4) and makes it boil, in other words, transforms it to steam.
4. This steam turns the turbine generator group which produces electricity,
5. The steam is cooled by the water cooling system or the tertian circuit: it is condensed i.e it becomes a liquid.

The risk of nuclear power plants usually comes from the used fuel, uranium.
Uranium is not classified among the most toxic elements of its radioactivity perspective, but in the reaction of nuclear fission it undergoes in the core, it transforms into different substances which are very radioactive.
Moreover, this reaction must be carefully controlled under penalty of race, which could lead to a catastrophic meltdown. Thus,it is important to study the behavior of a reactor's rated velocity(performance improvement) or accidental system (safety study) to simulate the flows of fluid within the reactor .

The STMF develops models and numerical methods dedicated to the study of flows in the primary and secondary circuits, the enclosure of nuclear reactors, as well as experimentation associated.
These flows generally comprise several phases, and it is important to understand and simulate their dynamics accurately.

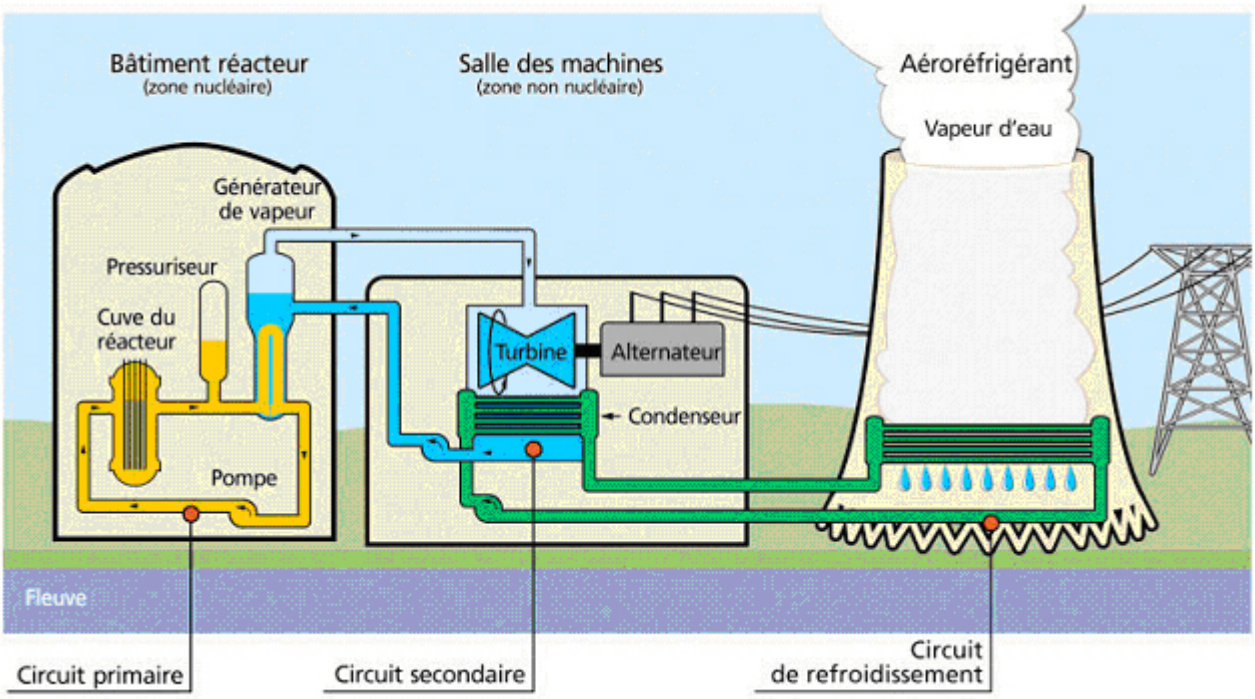

Figure 1.4: Working principle of a pressurized water reactor PWR

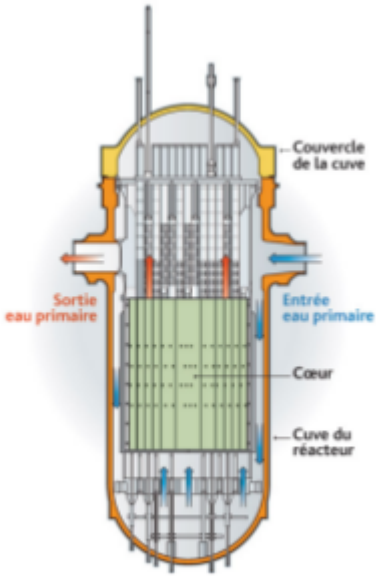

Figure 1.5: Diagram of a PWR900 vessel

As part of research in physics reactor, LMEC suggest models and numerical two-phase thermal-hydraulics codes (for models and numerical schemes) to simulate the static and dynamic behavior of nuclear core systems requiring a good knowledge of two-phase flows[6]

The ***FLICA 4*** code developed by LMEC is dedicated to the calculation of multiphase

6. see the definition of a two-phase flow in the introductory chapter 6

flows (Liquid and steam water) to permanent or transient state in core of WRP. This is a three-dimensional thermal-hydraulic code using the method of finite volume.

The two-phase flow comprising a liquid phase and a steam phase is modeled by four conservation equations, with a set of closure laws.

In ***FLICA 4*** computing code, the model with four equations is solved by VFRoe [2] numerical method on a collocated mesh with a patch to improve precision low Mach number. This correction allows better capture of the almost incompressible dynamics but but introduces numerical instabilities flows. introduces numerical instability flows These instabilities make certain calculations difficult to carry out, in the case of flows boiling in differentially heated parallel channel.

On this configuration, we will test two new numerical approaches. The first scheme is the use of an upwinding in the source terms in order to obtain a well balanced scheme. The second scheme is inspired by schemes on collocated mesh and precise for low Mach number.

Chapter 2

The mathematical model

A two-phase flow occurs in a system containing two different phases. For example, it is possible to study a flow with water as a vapor flow and as a liquid.

A two-phase flow is defined as a mixture of two phases where the topology, the composition and the exchange phenomenon are parameters which can permanently fluctuate. Because of their wide variety, they can be found among the industrial applications such as nuclear reactors, turboprops, $\cdots$

Since it is quite difficult to find a valid mathematical model which rules a two-phase system, this type of problem can be complex. These mathematical models are based both on universal laws (principle of mass conservation, law of conservation of momentum) as well as behavior laws which are essentially used to close the system of equations.
Even if it is always possible to write local equations, the direct solving seems to be almost unaffordable because of the small scales of the problem. Moreover, the modeling of the flows encountered in the industry need a particular approach (homogenization type)[1].

There are many two-phase flow applications, especially in the industry through nuclear engineering, petroleum engineering, chemical engineering and automobile and aerospace propulsion but also in biology and in chemistry through lubrication or distillation.

There are two main groups of two-phase flow modeling methods: the Lagrangian methods and the Eulerian methods. Within the Eulerian methods chosen at the STMF, the two-phase flow modeling is based on a homogenization process[1] [3]. These techniques consist in considering both phases as two distinct continuous media sharing the available volume and then solving the equations ruling each phase's state variables. In their most generic formulation, they result in open systems, characterized by two distinct velocities and pressures. Thus, if we consider only one space-dimension, these models are formed of at least six equations: two mass balance equations, two momentum equations, two energetic equations to which two closure relations may be added.

2.1 Eulerian single-phase balance equations

In the next part we will fix $k = v, l$ the respective indexes of each phase. Every two-phase medium can be assimilated to a set of pure one-phase areas in which the local equations of fluid mechanics are applicable. These areas are separated by infinitely thin interfaces and supposed without mass.

Setting the mass balance ρ_k, the momentum $\rho_k \mathbf{u}_k$ and total energy balance $\rho_k E_k = \rho_k e_k + \rho_k \frac{\mathbf{u}_k^2}{2}$ with $e_k(T_k)$ as the specific intern energies which depend on the temperatures T_k.
We obtain for each phase the classical system of the Navier-Stokes equations

$$\begin{array}{lllllll} \frac{\partial \rho_k}{\partial t} & + & \mathtt{div}(\rho_k \mathbf{u}_k) & & & = & 0 \\ \frac{\partial \rho_k \mathbf{u}_k}{\partial t} & + & \mathtt{div}(\rho_k \mathbf{u}_k \otimes \mathbf{u}_k) & + & \nabla p_k + \mathtt{div}(-\bar{\tau}_k) & = & \rho_k \mathbf{f}_k^v \\ \frac{\partial \rho_k E_k}{\partial t} & + & \mathtt{div}(\rho_k E_k + p_k)\mathbf{u}_k & + & \mathtt{div}(\mathbf{q}_k - \bar{\tau}_k.\mathbf{u}_k) & = & \rho_k \mathbf{f}_k^v.\mathbf{u}_k \end{array} \tag{2.1}$$

Where $\bar{\tau}_k$ is the tensor constraints on viscosity,$\mathbf{f}_k^v$ the set of external volumetric forces which apply to the phase k and q_k is the heat flow.

2.2 Averaged balance equations

To describe the evolutions of the variables in both fluids, we denote by $\Omega_k(t)$ the space domain taken by the phase k in the flow domain at time t, and we define X_k as each phase's characteristic function by:

$$X_k(x,t) = \begin{cases} 1 & if \quad x \in \omega_k(t) \\ 0 & else \end{cases}, \tag{2.2}$$

Thus, we have $\sum_k X_k(x,t) = 1$. We denote by W the velocity of the interface. The equation ruling the evolution of X_k is :

$$\frac{\partial}{\partial t} X_k = W \cdot n_k \cdot \delta, \tag{2.3}$$

Where n_k is the unit normal vector at the interface conducted from the fluid k to the fluid $\bar{k}$[1] and δ is the Dirac delta function concentrated on the interface.

On the other hand, we also have

$$\nabla X_k = -n_k \delta \tag{2.4}$$

Which allows us to write 2.3 as

$$\frac{\partial}{\partial t} X_k + W \cdot \nabla X_k = 0 \tag{2.5}$$

We may also remind that the function X_k checks the following identities as distributions:

1. if $k = g$ then $\bar{k} = l$ else $\bar{k} = g$

$$\nabla(X_k f) = X_k \nabla f + f \nabla X_k = X_k \nabla f - f n_k \delta$$
$$\mathtt{div}(X_k \mathbf{u}) = X_k \mathtt{div}\mathbf{u} + \mathbf{u} \nabla X_k = X_k \mathtt{div}\mathbf{u} - \mathbf{u} n_k \delta$$
$$\mathtt{div}(X_k \tau) = X_k \mathtt{div}\tau + \tau \nabla X_k = X_k \mathtt{div}\tau - \tau n_k \delta$$

Where f is a scalar function, $\mathbf{u}$ a vector, τ a second-order tensor. The equations 2.1 and 2.5 give a complete and highly detailed description of the flow. However, they cannot initialize a numerical method when the space steps are much more important than the size of the liquid and gaseous inclusions. To deal with these cases, we must consider the use of homogenized models.

In order to set the six-equation two-phase model or the four-equation drift model of ***FLICA 4*** [15], we use the methods described in [4] or [5] for the calculation of the average, as we mentioned in this chapter's introduction.
The homogenization process is given in [7], to which averaged interface conditions[7] must be added and also assuming that the tensor constraints on viscosity, the heat flow and the set of external volumetric forces equal their average value.
Thus, we obtain the following system of equations

$$\begin{aligned}
\frac{\partial}{\partial t}(\alpha_k \rho_k) + \mathtt{div}(\alpha_k \rho_k \mathbf{u}_k) &= \Gamma_k \\
\frac{\partial}{\partial t}(\alpha_k \rho_k \mathbf{u}_k) + \mathtt{div}(\alpha_k \rho_k \mathbf{u}_k \otimes \mathbf{u}_k) + \nabla(\alpha_k p_k) + \mathtt{div}(-\alpha_k \bar{\tau}_k) & \\
= \mathbf{M}_k^\Gamma + p_{kI} \nabla \alpha_k + \mathbf{F}_k^d + \alpha_k \rho_k \mathbf{f}_k^v & \qquad (2.6) \\
\frac{\partial}{\partial t}(\alpha_k \rho_k E_k) + \mathtt{div}(\alpha_k \rho_k E_k + \alpha_k p_k)\mathbf{u}_k + \mathtt{div}(\alpha_k(\mathbf{q}_k - \bar{\tau}_k \cdot \mathbf{u}_k)) & \\
= \mathbf{H}_k^\Gamma - p_{kI} \frac{\partial \alpha_k}{\partial t} + \mathbf{F}_k^d \cdot \mathbf{u}_{kI} + Q_{kI} + \alpha_k \rho_k \mathbf{f}_k^v \cdot \mathbf{u}_k &
\end{aligned}$$

with :

- Γ_k,$\mathbf{M}_k^\Gamma$,$\mathbf{H}_k^\Gamma$ are terms of mass interfacial transfer,
- Q_{kI} is the term of heat interfacial transfer,
- $\mathbf{F}_k^d$ is the term of interfacial friction force,
- p_{kI} and $\mathbf{u}_{kI}$ are respectively the pressure and the velocity of the phase k at the interface.

The system 2.6 reflects the general form of the two-fluid model obtained by the process of average. However, we can only count in this system six equations for seven unknowns

ρ_k, $\mathbf{u}_k$, E_k et α_k. This system is open and we have to find closure relations.
In the case of classical six-equation two-fluid models, this problem is solved thanks to the relation between the pressures p_v and p_l the common pressure assumption is widespread[4]. Indeed, supposing the equality of the phasic pressures is physically legitimate:

$$p_l = p_v = p_{vI} = p_{lI} = p$$

And also constitutive laws under the form $e_k = e_k(T_k)$ for the phasic intern energies. And therefore the conventional two-fluid model of six equations written as:

$$\begin{aligned}
\frac{\partial}{\partial t}(\alpha_k\rho_k) + \mathtt{div}(\alpha_k\rho_k\mathbf{u}_k) &= \Gamma_k \\
\frac{\partial}{\partial t}(\alpha_k\rho_k\mathbf{u}_k) + \mathtt{div}(\alpha_k\rho_k\mathbf{u}_k\otimes\mathbf{u}_k) + \alpha_k\nabla(p) + \mathtt{div}(-\alpha_k\bar{\tau}_k) & \\
= \mathbf{M}_k^{\Gamma} + p\nabla\alpha_k + \mathbf{F}_k^d + \alpha_k\rho_k\mathbf{f}_k^v & \qquad (2.7)\\
\frac{\partial}{\partial t}(\alpha_k\rho_k E_k) + \mathtt{div}(\alpha_k\rho_k E_k + \alpha_k p)\mathbf{u}_k + \mathtt{div}(\alpha_k(\mathbf{q}_k - \bar{\tau}_k\cdot\mathbf{u}_k)) & \\
= \mathbf{H}_k^{\Gamma} - p\frac{\partial\alpha_k}{\partial t} + \mathbf{F}_k^d\cdot\mathbf{u}_{kI} + Q_{kI} + \alpha_k\rho_k\mathbf{f}_k^v\cdot\mathbf{u}_k &
\end{aligned}$$

2.3 Obtaining the equations of the drift model

By making simplistic assumptions on the velocity and the temperature of the existing phases, it is possible to move from a six-equation model to a four-equation model. The two-phase flow, which contains a liquid phase and a vapor phase, is modeled by four balance equations:

- Conservation of total mass of the mixture
- Vapor mass balance
- Momentum of mixture balance
- Energy of mixture balance

This model, called four-equation drift, is based on very simplistic assumptions.

- First, the velocity gap between the phases is *a priori* ruled by a correlation $\mathbf{u}_r = \mathbf{u}_v - \mathbf{u}_l = f_r(c_v, \mathbf{u}_m, \rho_m)$.
  This assumption limits the use of the model to the small velocity deviations between both phases and most of the correlations suppose that there are flows which have almost the same velocity.

- Secondly, the vapor is supposed to reach the saturation level: $T_v = T_v^{sat}$.
  This assumption avoids particularly the simulation of overheated vapor.

2.3.1 The drift model of *FLICA 4*

FLICA 4 is a 3D two-phase thermal-hydraulics software which is developed at the STMF service of the CEA. It is dedicated to the calculations of two-phase flows (liquid water and steam) within the core of the nuclear reactors for unsteady and steady flows. It considers the presence of solid hindrances within the flow, through a porous perspective.

The mathematical model of ***FLICA 4*** is a four-equation model which is established from the six-equation model (cf. 2.7), by including the porosity, by removing two partial differential equations and by replacing them by two algebraic equations which give the velocity gap between the phases and the thermodynamic imbalance.

A Drift model is used to take into account the shift between the two phases (vapor and liquid) and therefore, gives the velocity gap $\mathbf{u}_r = \mathbf{u}_v - \mathbf{u}_l = f_r(c_v, \mathbf{u}_m, \rho_m)$.
Both phases are compressible and supposed to be at the same pressure.
The thermodynamic imbalance between the phases is obtained by supposing that one of them (vapor in our case) reaches the saturation level.

The conservation of mass of the mixture

It is obtained by adding the liquid and vapor mass balance equations(2.7). The linearity of the operators in these two equations allows us to write to following equation:

$$\frac{\partial}{\partial t}\phi(\rho_l\alpha_l + \rho_v\alpha_v) + \mathtt{div}\phi(\rho_l\alpha_l\mathbf{u}_l + \rho_v\alpha_v\mathbf{u}_v) = 0$$

The l index represents the liquid phase and the v index the vapor phase.

The vapor mass balance equation

$$\frac{\partial}{\partial t}(\phi\rho_v\alpha_v) + \mathtt{div}\phi(\rho_v\alpha_v\mathbf{u}_v + K_{c_v}\mathtt{grad}c_v) = \phi\Gamma_v$$

Γ_v : vapor mass created by unit of time
K_{cv} is an optional turbulent diffusion coefficient

The momentum balance equation

The momentum balance equation for the mixture of the two phases is also obtained by adding the liquid and vapor momentum balance equations in (2.7) and is written:

$$\frac{\partial}{\partial t}\phi(\rho_v\alpha_v\mathbf{u}_v + \rho_l\alpha_l\mathbf{u}_l) + \mathtt{div}\phi(\textstyle\sum_{k=v,l}\rho_k\alpha_k\mathbf{u}_k \otimes \mathbf{u}_k + \alpha_k p\mathbf{I} + \alpha_k\Pi_k) = \phi\rho\vec{g} + \phi\vec{\tau}$$

$\mathbf{I}$: is the identity matrix with the same dimension as $\mathbf{u}$
Π_k : is the tensor constraints on viscosity of the phase k

The energy balance equation

The energy balance equation is written :

$$\begin{array}{c}\frac{\partial}{\partial t}(\phi \sum_{k=v,l} \rho_k \alpha_k E_k) + \texttt{div}\phi(\sum_{k=v,l} \rho_k \alpha_k \mathbf{u}_k E_k - \alpha_k \Pi_k \mathbf{u}_k - q_k) \\ = Q_{ktot} + \phi \left(\sum_{k=v,l} \rho_k \alpha_k \mathbf{u}_k\right) \cdot \vec{g}\end{array}$$

Q_{ktot} : is the source term of total volumetric power received by the phase k.
q_k : is the thermal flow generated by the thermal conduction of the phase k

Thus the four-equation ***FLICA 4*** model is written under the form:

$$\begin{array}{ccccc}
\frac{\partial}{\partial t}\phi(\rho_l\alpha_l + \rho_v\alpha_v) & + & \texttt{div}\phi(\rho_l\alpha_l\mathbf{u}_l + \rho_v\alpha_v\mathbf{u}_v) & = & 0 \\
\frac{\partial}{\partial t}(\phi\rho_v\alpha_v) & + & \texttt{div}\phi(\rho_v\alpha_v\mathbf{u}_v + K_{cv}\texttt{grad}cv) & = & \phi\Gamma_v \\
\frac{\partial}{\partial t}\phi(\rho_v\alpha_v\mathbf{u}_v + \rho_l\alpha_l\mathbf{u}_l) & + & \texttt{div}\phi(\sum_{k=v,l}\rho_k\alpha_k\mathbf{u}_k \otimes \mathbf{u}_k + \alpha_k p\mathbb{I} + \alpha_k\Pi_k) & = & \phi\rho\vec{g} + \phi\vec{\tau} \\
\frac{\partial}{\partial t}(\phi \sum_{k=v,l} \rho_k\alpha_k E_k) & + & \texttt{div}\phi(\sum_{k=v,l} \rho_k\alpha_k\mathbf{u}_k E_k - \alpha_k\Pi_k\mathbf{u}_k - q_k) & & \\
 & = & Q_{tot} + \phi \left(\sum_{k=v,l} \rho_k\alpha_k\mathbf{u}_k\right) \cdot \vec{g} & &
\end{array} \tag{2.8}$$

Source terms of the *FLICA 4* model

Total mass equation :
The first component is null because there isn't any material created but only a transformation from liquid to vapor.

Vapor mass equation :
Γ_v is the phase change rate: quantity of liquid vaporized by unit of time, which is the sum of the vaporization on contact with the heating walls Γ_{wv} and the mass exchange at the interface between the phases Γ_{lv}.

The phase change term Γ_v is non-zero if the enthalpy h_m of the fluid lies between the enthalpy of the liquid and the enthalpy of the vapor. However, ***FLICA 4*** also models the undersaturated boiling where the bubbles appear only on contact with the heated walls whereas the average temperature in the canal stays lower than the boiling temperature

Momentum equation :
The source term τ represents the sum of the mixture frictions on the walls τ_w and on the singular hindrances τ_s (the solid hindrances present in the reactor).
The parietal friction term is linearized under the form $-K\rho_m\mathbf{u}_m$ with K defined as a constant normally depending on the hydraulic diameter and the Reynolds number. Therefore, we will not consider in this case the heating wall coefficient and the two-phase multiplier either

Energy equation :
The source of energy provided to the fluid is described by the Q_{tot} term, which is the volumetric power received by the fluid which comes into contact with the heating walls.

2.3.2 The Drift model in CoreFlows

In order to replicate the numerical complexities of ***FLICA 4***, we are going to develop a replica of the code based on a simplified version of the model, which will allow us to study new numerical approaches.

We shall consider simplified source terms of phase shift, parietal friction and heating.

Rather than supposing that the steam reaches its saturation level, we may suppose the thermal equilibrium: $T_v = T_l$.
We have seen that this assumption is simpler and more general than the vapor saturation assumption. As of now, we will not take into account the relative velocity u_r.

Therefore, we are working on a four-non-linear partial differential equation system which conservative form is the following:

$$\partial_t U + \nabla \cdot F(U) = S, \tag{2.9}$$

with the following terms for the vector of conservative variables U :

$$U = \begin{pmatrix} \alpha_v\rho_v + \alpha_l\rho_l \\ \alpha_v\rho_v \\ \alpha_v\rho_v\mathbf{u}_v + \alpha_l\rho_l\mathbf{u}_l \\ \alpha_v\rho_v E_v + \alpha_l\rho_l E_l \end{pmatrix},$$

And the matrix of flow $F(U) = F_{NV}(U) + F_V(U)$

$$F_{NV}(U) = {}^t\begin{pmatrix} \alpha_v\rho_v{}^t\mathbf{u}_v + \alpha_l\rho_l{}^t\mathbf{u}_l \\ \alpha_v\rho_v{}^t\mathbf{u}_v \\ \alpha_v\rho_v\mathbf{u}_v \otimes \mathbf{u}_v + \alpha_l\rho_l\mathbf{u}_l \otimes \mathbf{u}_l + p\mathbb{I}_d \\ \alpha_v\rho_v H_v{}^t\mathbf{u}_v + \alpha_l\rho_l H_l{}^t\mathbf{u}_l \end{pmatrix}, \tag{2.10}$$

$$F_V(U) = {}^t\begin{pmatrix} 0 \\ 0 \\ \alpha_v\mu_v\vec{\nabla}\mathbf{u}_v + \alpha_l\mu_l\vec{\nabla}\mathbf{u}_l \\ \alpha_v\mu_v\mathbf{u}_v \cdot \vec{\nabla}\mathbf{u}_v + \alpha_l\mu_l\mathbf{u}_l \cdot \vec{\nabla}\mathbf{u}_l + \alpha_v\lambda_v\vec{\nabla}T + \alpha_l\lambda_l\vec{\nabla}T \end{pmatrix}. \tag{2.11}$$

With, for each phase:
The viscosity $\mu_k, k = l, v$
The conductivity $\lambda_k, k = l, v$
The total energy: $E_k = e_k + \frac{1}{2}|\mathbf{u}_k|^2, k = l, v$
The total enthalpy: $H_k = h_k + \frac{1}{2}|\mathbf{u}_k|^2, k = l, v$
Where e_k is the intern energy, and $h_k = e_k + \frac{p}{\rho k}$ the enthalpy associated to the phase k

State laws and constitutive parameters

For each phase, the state low is approached by a linear stiffened gas law

$$p_k = (\gamma_k - 1)\rho_k e_k - \gamma_k p_{0k}$$

And a linear intern energy law

$$e_k(T) = e_k(345K) + c_{vk}(T - 345K)$$

Which are applicable in the vicinity of the saturation point $p_0 = 155bars, T_0^{sat} = 345$.

The values of density, constant-volume intern energy and heat capacity, viscosity and conductivity for each phase around the point (p_0, t_0) are derived from the NIST base [20].

Source terms

We shall consider a simplified source term under the form

$$S = \begin{pmatrix} 0 \\ \Gamma_v(c_v, T) \\ \rho_m \vec{g} - K\rho_m ||\mathbf{u}_m|| \mathbf{u}_m \\ \Phi + \rho_m \vec{g} \cdot \mathbf{u}_m - K\rho_m ||\mathbf{u}_m||^3 \end{pmatrix},$$

with

$$\begin{aligned} \rho_m &= \alpha_l \rho_l + \alpha_v \rho_v \\ \mathbf{u}_m &= \frac{\alpha_l \rho_l \mathbf{u}_l + \alpha_v \rho_v \mathbf{u}_v}{\alpha_l \rho_l + \alpha_v \rho_v} \\ h_m &= \frac{\alpha_l \rho_l h_l + \alpha_v \rho_v h_v}{\alpha_l \rho_l + \alpha_v \rho_v} \end{aligned}$$

The first component is null because there isn't any material created but only a transformation from liquid to vapor.
Γ_v is the phase change rate: quantity of liquid vaporized by unit of time.
$\rho_m \vec{g} - K\rho_m ||\mathbf{u}_m|| \mathbf{u}_m$ represents the volumetric forces applied to the fluid. $\vec{g}$ represents the

gravity and $-K\rho_m||\mathbf{u}_m||\mathbf{u}_m$ the friction on the solid hindrances in the reactor. Φ represents the thermal power transferred to the liquid by unit of time.
The phase change term Γ_v is non-zero if the enthalpy h_m of the fluid lies between the enthalpy of the liquid and the enthalpy of the vapor, both taken at the saturation point (p0,t0) (p_0, T_0).

In this case, all the heat received by the thermal flow is used to the phase change:

$$\Gamma_v = \begin{cases} \frac{\Phi}{\mathcal{L}} & \text{si } h_l^0 \leq h < h_v^0 \text{ et } c_v < 1 \\ 0 & \text{sinon} \end{cases} . \tag{2.12}$$

If the thermal power is null, $\Gamma_v = 0$ and there is no phase change.
Therefore, we are not modeling a spontaneous boiling/condensation of the fluid, and the phased are never supposed to reach the saturation level.

The parietal friction term is linearized under the form $-K\rho_m||\mathbf{u}_m||\mathbf{u}_m$ with K defined as a constant normally depending on the hydraulic diameter and the Reynolds number. Therefore, we will not consider in this case the heating wall coefficient and the two-phase multiplier either

Chapter 3

Numerical method

We are going to apply a finite volume method in order to solve the balance equations ruling the two-phase flow.
The exact solution of the problem is approached by a grid constant function. Its values are solutions of a discrete problem obtained by integrating local equations on each mesh element in the domain of definition. The conservation equations of the thermohydraulic model are written under the following form:

$$\partial_t U + \texttt{div} \cdot F(U) = S(U) \tag{3.1}$$

where
- U is the vector of conservative variables
- $F(U)$ is the flow vector in the three directions. It can be separated into two parts:

$$F(U) = F_{NV}(U) + F_V(U)$$

such that

- $F_{NV}(U)$ represents the convective flows,
- $F_V(U)$ represents the contribution of the viscous flows,
- $S(U)$ is the vector of the source terms .

We may outline the specific notations of the mesh:
The footprint of a control cell on a horizontal plane is a quadrangle or a random triangle. The control volumes are numbered by the index i, as well as the constant variables.

We use the following rules for the indexes:

- i is the index of a cell of C_i
- j is the index of the neighbor of C_i
- ij is the interface between neighbor cells C_i and C_j
- $I(i)$ is the set of indexes j whose neighbors of C_j are neighbors of C_i

- $V(i)$ is the set of indexes j such that the cell C_j has a common edge with the cell C_i

We denote $Vol(Ci)$ as the volume of the control cell C_i et ∂C_i its border. We also denote S_{ij} the area of the interface $\partial C_i \cap \partial C_j$.

The finite volume formulation consists in associating a constant value U_i for the vector of conservative variables to each control cell C_i, and then integrating the conservation equations 3.1 on this cell. We denote $\partial\Omega$ the border of the domain Ω and $\mathtt{n} = (n_x, n_y, n_z)$ the outgoing normal unit vector of $\partial\Omega$

The semidiscrete finite volume formulation of 3.1 is :

$$\int_{C_i} S(U)\,\mathrm{d}v^8 = Vol(C_i)\partial_t U_i + \int_{\partial C_i} F(U)\cdot\mathtt{n}\,\mathrm{d}\sigma^9 \tag{3.2}$$

We can see in the appendix the time discretization (explicit and implicit schemes) and the processing of the boundary conditions.

3.1 Finite volume formulation of *FLICA 4*

Within ***FLICA 4***, the 3D meshes considered are obtained by a translation following the axial direction (z direction) of a 2D mesh (structured or not), in a horizontal plane (xy plane).

3.1.1 Convective flows

We are looking at the calculation of the convective flows shown in the equation 3.2.
The area of the cell C_i is formed of polygonal faces:

$$\int_{\partial C_i} F_{NV}(U)\cdot\mathtt{n}\,\mathrm{d}\sigma = \sum_{j\in I(i)} \int_{\partial C_{ij}} F_{NV}(U)\cdot\mathtt{n}\,\mathrm{d}\sigma + \int_{\partial C_i\cap\partial\Omega} F_{NV}(U)\cdot\mathtt{n}\,\mathrm{d}\sigma \tag{3.3}$$

In the first term, the sum is extended to all the neighbors C_j which have a common interface $\partial C_{ij} = \partial C_i \cap \partial C_j$ with the cell C_i. The second term represents the integration at the boundaries. The intern flows, which are the reflect of an interface strictly lower to the domain Ω, are approached by :

$$\int_{\partial C_{ij}} F_{NV}(U)\cdot\mathtt{n}\,\mathrm{d}\sigma = S_{ij}\tilde{F}(U_i, U_j, \mathtt{n}_{ij}) \tag{3.4}$$

where :
$\tilde{F}$ is the numeric flow that we will express further.

8. représente un élément de volume
9. représente un élément de surface

The limited flows reflect an interface on the boundary of the domain Ω, and are calculated thanks to the same numeric flow function, and after introducing a fake adjacent cell to the face C_{ik} outside the domain (7.2).

VFRoe Scheme

FLICA 4 currently uses a VFRoe scheme with a pressure correction to make it more accurate. These equations are discretized thanks to the finite volume method.
The domain is divided into cells (also called mesh elements) which can have any form (non-structured mesh and/or non-conform mesh). The following releases of ***FLICA 4*** use the alternative called VFRoe [2], which consists in calculating the state at the interface (U^*) and then taking the physical flow of this state :

$$\tilde{F}(U_i, U_j, \mathtt{n}_{ij}) = F(U^*)$$

With

$$U^*(U_i, U_j, \mathtt{n}_{ij}) = \frac{U_i + U_j}{2} - \mathtt{signe}(A_{ij})\frac{U_i - U_j}{2}. \tag{3.5}$$

pressure correction In the alternative VFRoe of the scheme (equation 3.5), the pressure correction consists in calculating the primitive state V^* associated to U^* (see [6]), then replacing the pressure of V^* by $\frac{p_i+p_j}{2}$ and finally calculating a new conservative stage $\frac{p_i+p_j}{2}$

Unfortunately, although the scheme's accuracy is upgraded using this method, these pressure corrections downgrades the stability of the original Roe scheme (oscillations of the solution) (see figure 3.1)

3.1.2 Viscous flows

We are now going to work on the calculation of the second order terms associated to the modeling of the viscous constraints of the two-phase fluid studied. The viscous contribution of the total flow $F(U)$ is a function of the conservative variables U and the gradient ∇U of these variables. We can write again the viscous flows which occur in the equation 3.2 under the following form :

$$\int_{\partial C_i} (F_V(U)\mathtt{n}_x + G_V(U)\mathtt{n}_y + H_V(U)\mathtt{n}_z)\mathtt{n}\,\mathrm{d}\sigma = \int_{\partial C_i} \tilde{F}_V(U, \nabla U, \mathtt{n})\,\mathrm{d}\sigma \tag{3.6}$$

where $\tilde{F}_V(U, \nabla U, \mathtt{n})$ represents the viscous flow in the direction normal to the surface ∂C_i. This last expression can be divided into two terms involving the internal interfaces and on the boundary:

$$\int_{\partial C_i} \tilde{F}_V(U, \nabla U, \mathtt{n})\,\mathrm{d}\sigma = \sum_{j\in I(i)} \int_{\partial C_{ij}} \tilde{F}_V(U, \nabla U, \mathtt{n})\,\mathrm{d}\sigma + \sum_{k\in K(i)} \int_{\partial C_{ik}} \tilde{F}_V(U, \nabla U, \mathtt{n})\,\mathrm{d}\sigma \tag{3.7}$$

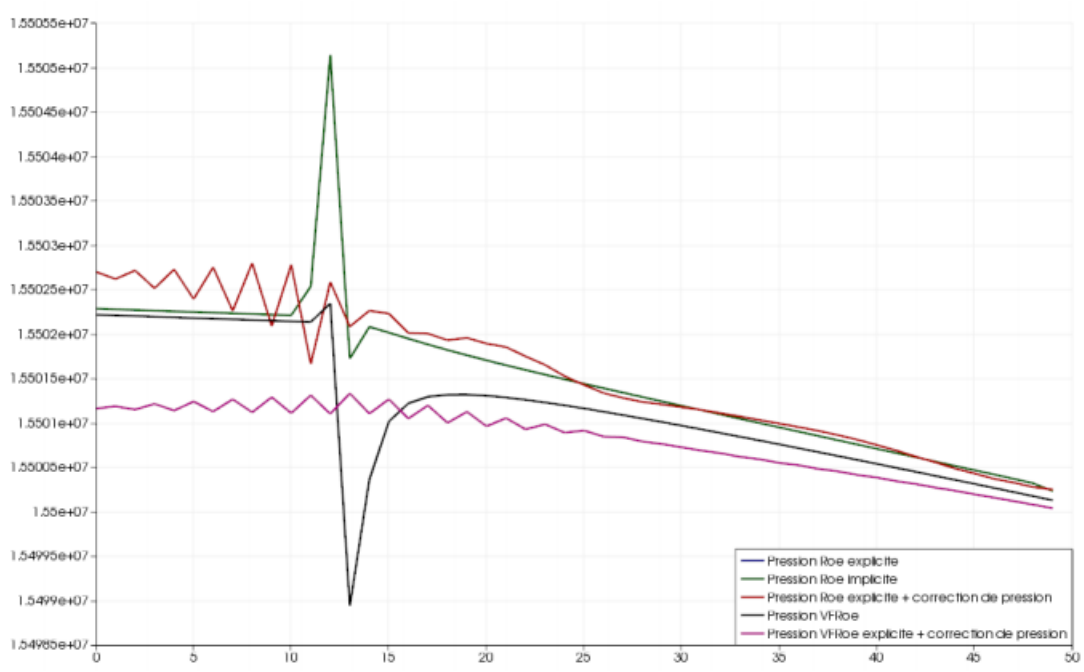

Figure 3.1: Comparison of numerical schemes: stationary pressure $U_{entree} = 1m/s$

To calculate the viscous flows at an interface ∂C_{ij}, it is necessary to evaluate the gradient of conservative variables at this interface. Actually, we can notice that the viscous flow is a linear function of the velocity gradient and each phase's enthalpy. In a structured mesh, the estimation of these quantities' gradient at an interface is classical with a finite difference central scheme.

In the case of a non structured mesh, the estimation of the gradient at the interface is not as classical.

To approach properly the partial derivatives (normal and tangential) of U on an interface depending on the values at the barycenter of the mesh, we will introduce the following points (see Figure 3.2):

- N_1 and N_2 two points belonging to the interface C_{ij}
- N_G and N_D two points belonging to the interface C_{ij} passing through the midpoint M

Then, we define the normal and tangential derivatives of U by :

$$\frac{U_{N_2} - U_{N_1}}{||N_1 N_2||} \text{ and } \frac{U_{N_D} - U_{N_G}}{||N_D N_G||}$$

These definitions are first-order consistent approximations of the exact derivatives at the midpoint M if U_{N_1}, U_{N_2}, U_{N_G}, U_{N_D} are second-order approximations of the vector of conservative variables respectively at points N_1, N_2, N_G and N_D.

The geometric definition of these points as well as the detailed calculations of the gradient at an interface are all given in the document [16]

This approach leads to an estimation of the viscous flow combining the control cell C_i and all the neighbors which have at least one common vertex.

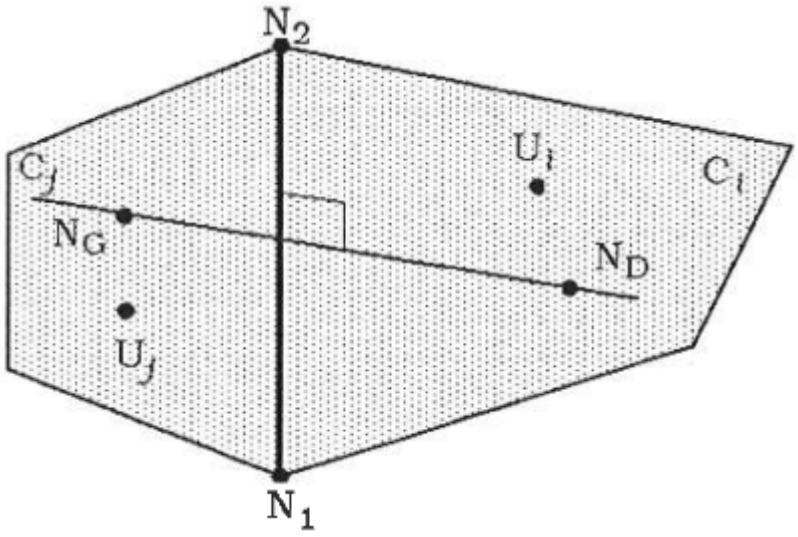

Figure 3.2: Calculation of the gradients for the viscous flows

3.1.3 Discretization of the source terms

The source terms are calculated using a central scheme. Since the conservative variables are constant in the control cell C_i, we obtain the following discretization for the second member of the equation 3.2 :

$$\int_{C_i} S(U), \mathrm{d}v \approx vol(C_i)S(U_i).$$

3.2 Finite volume formulation of CoreFlows

As a result of the final year project we realized in February 2015, we have developed a model which compares the Roe schemes and the VFRoe schemes with and without pressure correction on calculations of 1D and 2D boiling channels. In conclusion of this work, we discovered that it is essential shift the center of the Roe schemes in order to apprehend the solution efficiently. Theoretical guidelines inspired this processing of the source terms in the case of stiff or discontinuous source terms. We are going to extend this work by comparing the balanced Roe scheme with specific schemes with low Mach number in the CoreFlows environment on CDMATH and Salomé.

3.2.1 Convective flows

The convective flows are under the form

$$\tilde{F}(U_i, U_j, \mathtt{n}_{ij}) = \frac{F(U_i) + F(U_j)}{2}\mathtt{n}_{ij} - D_{ij}\frac{U_i - U_j}{2}. \tag{3.8}$$

where D_{ij} is the shift matrix of the scheme

Upwind scheme

The numerical flow of the Roe scheme ([21]) is based on the of a one-dimensional linearized Riemann's problem in the direction normal to an interface between two control cells

$$\partial_t U + A(U_i, U_j)\partial_{x_n} U = 0$$

With:

- x_n represents the direction of the normal $\mathtt{n}$,such that $U(x_n, 0) = \begin{cases} U_i^t & \text{si } x_n < 0 \\ U_j^t & \text{si } x_n > 0 \end{cases}$,
- $A(U_i, U_j)$ A(Ui,Uj) linearized jacobian matrix, called Roe matrix (see appendix 7.3)

The numerical flow obtained thanks to the exact resolution of this Riemann's problem, which is also the flow that the first releases of ***FLICA 4*** use is given by the following

expression

$$\tilde{F}(U_i, U_j, \mathtt{n}_{ij}) = \frac{F(U_i) + F(U_j)}{2}\mathtt{n}_{ij} - A_{ij}\frac{U_i - U_j}{2}. \tag{3.9}$$

The detail of the calculation of the Roe matrix and the shift are given in the appendix 7.3.

The first term of (3.9 and 3.5) is the central part which is responsible for the consistency of the numerical flow $\tilde{F}$ with the physical flow $F : \tilde{F}(U, U, \mathtt{n}) = F(U)\mathtt{n}$ because the other terms get cancelled when $U_i = U_j$.

The last terms of (3.9 and 3.5) ensure the stability of the numerical scheme and some other numerical properties such as the maximum principle or the positivity of certain physical quantities such as concentration, density or pressure.

Low Mach scheme

Although the Roe schemes are stables, they may have accuracy problems with low Mach number [17]. When the Mach number of a compressible flow is low and if there is no source term, the flow becomes nearly incompressible and the Godunov methods, which use a shift based on the acoustic wave propagation, become less accurate. To solve this problem of accuracy, the designers of ***FLICA 4*** have then introduced "*pressure corrections* in order to increase the accuracy.
The adjusted flow associated to the original formulation of the Roe scheme (equation 3.9) is written:

$$\tilde{F}(U_i, U_j, \mathtt{n}_{ij}) = \frac{F(U_i) + F(U_j)}{2}\mathtt{n}_{ij} - |A_{ij}|\frac{U_i - U_j}{2} + \frac{\tilde{a}(\rho_{mi}\mathbf{u}_{mi} - \rho_{mj}\mathbf{u}_{mj}) \cdot \mathtt{n}_{ij}}{2}\begin{pmatrix} 0 \\ 0 \\ \mathtt{n}_{ij} \\ 0 \end{pmatrix}, \tag{3.10}$$

where $\tilde{a}$ is the velocity of sound on the interface between U_i and U_j The detail of the calculation of the pressure correction is given in the appendix 7.3.5.

Staggered scheme

The new scheme is designed to be accurate with low Mach number in the spirit of the donor cell schemes ([18]). The method is necessarily implicit with a flow under the form:

$$F_{i+\frac{1}{2}} \quad = \quad \frac{F(U_i) + F(U_{i+1})}{2} + D_{MAC}\frac{U_i - U_{i+1}}{2}. \tag{3.11}$$

where D_{MAC} is built like the jacobian of F on the face except that the coefficients linked to the pressure are multiplied by -1. This idea represents the fact that for MAC schemes,

the scalar unknowns (density, pressure and energy) are discretized on mesh which is different from the one holding the velocity vector.

More specifically, the jacobian of the four-equation model under the assumption $\mathbf{u}_v = \mathbf{u}_l$ is in 1D

$$Jac(U) = \begin{pmatrix} \text{“}0 & 0 & {}^t\mathtt{n}_{ij} & 0 \\ -u_n c & u_n & c^t\mathtt{n}_{ij} & 0 \\ (\chi + \frac{1}{2}\kappa|\mathbf{u}|^2)\mathtt{n}_{ij} - u_n\mathbf{u} & \xi\mathtt{n}_{ij} & \mathbf{u}\otimes\mathtt{n}_{ij} + u_n\mathbb{I}_d - \kappa\mathtt{n}_{ij}\otimes\mathbf{u} & \kappa\mathtt{n}_{ij} \\ (\chi + \frac{1}{2}\kappa u^2 - H)u_n & \xi u_n & H^t\mathtt{n}_{ij} - \kappa u_n{}^t\mathbf{u} & (\kappa+1)\tilde{u}_n \end{pmatrix}$$

with χ, ξ and κ the partial derivatives of the pressure defined by the relation

$$\mathrm{d}p = \chi \mathrm{d}\rho + \xi \mathrm{d}(\rho c) + \kappa \mathrm{d}(\rho e).$$

and the matrix D_{MAC} related is:

$$D_{MAC}(U) = \mathtt{signe}(\mathbf{u}_n) \begin{pmatrix} 0 & 0 & {}^t\mathtt{n}_{ij} & 0 \\ -u_n c & u_n & c^t\mathtt{n}_{ij} & 0 \\ -(\chi + \frac{1}{2}\kappa|\mathbf{u}|^2)\mathtt{n}_{ij} - u_n\mathbf{u} & -\xi\mathtt{n}_{ij} & \mathbf{u}\otimes\mathtt{n}_{ij} + u_n\mathbb{I}_d + \kappa\mathtt{n}_{ij}\otimes\mathbf{u} & -\kappa\mathtt{n}_{ij} \\ (-\chi - \frac{1}{2}\kappa u^2 - H)u_n & -\xi u_n & H^t\mathtt{n}_{ij} + \kappa u_n{}^t\mathbf{u} & (-\kappa+1)\tilde{u}_n \end{pmatrix}.$$

3.2.2 Viscous flow F_V

Since our main goal is the improvement of the discretization of the convective flows and other source terms of ***FLICA 4***, the viscous flows of CoreFlows are discretized by a simple two-point approach which is less accurate than the ***FLICA 4*** one.

The gradient of U on the face between the cells C_i and C_j of the figure 3.2 is simply equal to:

$$\frac{U_i - U_j}{||G_i G_j||}$$

where G_i and G_j are the respective barycentres of the cells C_i and C_j.

3.2.3 Processing of the source terms

The semi-discrete finite volume scheme of the equation (3.1) is written under the form:

$$\frac{\mathrm{d}U_i}{\mathrm{d}t}(t) + \frac{1}{|C_i|}\sum_{j\in\nu(i)} s_{ij}F_{ij} = S_i \tag{3.12}$$

The central processing consists in taking

$$S(U_i) = S(U_i, x_i, t). \tag{3.13}$$

The upwinding (shift) processing consists in taking

$$S_i = \frac{1}{perimeter(i)} \sum_{j \in \nu(i)} s_{ij} S_{ij} \tag{3.14}$$

With

$$S_{ij} = \frac{S(U_i) + S(U_j)}{2} + \texttt{signe}(A_{i,j}) \frac{S(U_i) - S(U_j)}{2} \tag{3.15}$$

Chapter 4

Application : CoreFlows

4.1 Presentation of CoreFlows

CoreFlows is an open source C++/*Python* library intended to solve PDE systems arising from the thermalhydraulics of two phase flows in power plant boilers. It is a simple environment meant for students and researchers to test new numerical methods on general geometries with unstructured meshes. It proposes a few basic models and finite volume numerical methods.
Some of the main objectives are the study of

- Numerical schemes for compressible flows at low Mach numbers
- Well balanced schemes for stiff source terms (heat source, phase change, pressure losses)
- Flow inversion and counter-current two phase flows
- Schemes that preserve the phasic volume fraction $\alpha \in [0, 1]$
- Convergence of finite volume methods
- New preconditioners for implicit methods for two phase flows
- The coupling of fluid models or multiphysics coupling (eg thermal hydraulics and neutronics or thermal hydraulics and solid thermics)

CoreFlows relies on the toolbox [9] of the project CDMATH [8] for the handling of meshes and fields, and on the library Petsc [31] (version 3.4.5) for the handling of large sparse matrices.

4.2 The physical models

The physical models proposed in CoreFlows, are presented in order of mathematical complexity:

4.2.1 Scalar models

The transport equation

$$\partial_t h + \vec{u} \cdot \vec{\nabla} h = \Phi + \lambda_{sf}(T_s - T) \tag{4.1}$$

where

- h the main unknown is the fluid enthalpy field
- $\vec{u}$ is the constant transport velocity,
- Φ is the heat source term if explicitely known,
- T_s is the solid temperature field,
- $T = T_0 + \frac{H - H_0}{c_p}$ is the fluid temperature field
- λ_{sf} is the fluid-solid heat transfer coefficient,
- c_p is the fluid specific heat.

The class `TransportEquation` implements a scalar advection equation for the enthalpy of a fluid. The fluid can be either steam or liquid water around 1 bar or 155 bars.

In Examples's section (4.7.2), we find *Python*'s script to solve **??**.

The diffusion equation

The diffusion equation solved in CoreFlows is :

$$\partial_t T = d \triangle T + \frac{\Phi + \lambda_{sf}(T_f - T)}{\rho c_p} \tag{4.2}$$

where

- T the main unknown is the solid temperature field
- λ is the solid thermal conductivity,
- ρ is the solid density assumed constant,
- c_p is the solid specific heat,
- $d = \frac{\lambda}{\rho c_p}$ is the solid diffusivity

- Φ is the heat source term if explicitely known
- T_f is the fluid temperature field provided by the user

The class DiffusionEquation implementing a scalar diffusion equation for the temperature in a solid. The default values for ρ, c_p, λ are those of Uranium oxyde around $900K$.

4.3 The Navier-Stokes equations

The model consists of the following three balance laws for the mass, the momentum and the energy:

$$\left\{\begin{array}{lllllll} \frac{\partial \rho}{\partial t} & + & \nabla.\vec{q} & & & = & 0 \\ \frac{\partial \vec{q}}{\partial t} & + & \nabla.\left(\vec{q}\otimes\frac{\vec{q}}{\rho}+p\mathbb{I}_d\right) & - & \nu\Delta\vec{u} & = & \rho\vec{g}-K\rho||\vec{u}||\vec{u} \\ \frac{\partial(\rho E)}{\partial t} & + & \nabla.\left[(\rho E+p)\frac{\vec{q}}{\rho}\right] & - & \lambda\Delta T & = & \Phi+\rho\vec{g}\cdot\vec{u}-K\rho||\vec{u}||^3 \end{array}\right. \tag{4.3}$$

where

- ρ is the density,
- $\vec{u}$ the velocity,
- $\vec{q}=\rho\vec{u}$ the momentum,
- p the pressure,
- ρe the internal energy,
- $\rho E=\rho e+\frac{||\vec{q}||^2}{2\rho}$ the total energy,
- T the absolute temperature,
- Φ a heat source term,
- ν the viscosity and
- λ the thermal conductivity.

We close the system (4.3) by the ideal gas law $p=(\gamma-1)\rho e$ for steam water and a stiffened gas law $p=(\gamma-1)\rho e-\gamma p_0$ for liquid water. For the sake of simplicity, we consider constant viscosity and conductivity, and neglect the contribution of viscous forces in the energy equation.
The parameters $\lambda, \nu, \vec{g}, K$ and Φ can be set by the user.

As for Transport equation, we find a script's Python example to solve 4.3 in (4.7.2) section.

4.4 Two phase flow models

We present the homogeneised two phase flow models implemented in CoreFlows. This models are obtained by averaging the balance equations for each separated phase or for the mixture, using space, time or ensemble averaged quantities (see [1] and [3]). The drift model is used in the thermal hydraulics software Flica 4 (see [15]), whilst the two-fluid models are used in Cathare [18], Neptune_CFD [19], Cobra-TF [12], Relap5 [11].

The Drift model

The drift model is a system of four nonlinear equations taking the following conservative form:

$$\begin{cases} \partial_t(\alpha_g\rho_g + \alpha_l\rho_l) & +\nabla\cdot(\alpha_g\rho_g{}^t\vec{u}_g + \alpha_l\rho_l{}^t\vec{u}_l) & = 0 \\ \partial_t(\alpha_g\rho_g) & +\nabla\cdot(\alpha_g\rho_g{}^t\vec{u}_g) & = \Gamma_g(h_m,\Phi) \\ \partial_t(\alpha_g\rho_g\vec{u}_g + \alpha_l\rho_l\vec{u}_l) & +\nabla\cdot(\alpha_g\rho_g\vec{u}_g\otimes\vec{u}_g + \alpha_l\rho_l\vec{u}_l\otimes\vec{u}_l + p\mathbb{I}_d) & = \rho_m\vec{g} - K_g\alpha_g\rho_g||\vec{u}_g||\vec{u}_g - K_l\alpha_l\rho_l||\vec{u}_l||\vec{u}_l \\ \partial_t(\alpha_g\rho_g E_g + \alpha_l\rho_l E_l) & +\nabla\cdot(\alpha_g\rho_g H_g{}^t\vec{u}_g + \alpha_l\rho_l H_l{}^t\vec{u}_l) & = \Phi + \rho\vec{g}\cdot\vec{u} - K_g\alpha_g\rho_g||\vec{u}_g||^3 - K_l\alpha_l\rho_l||\vec{u}_l \end{cases}$$

where the total energy and total enthalpy are defined by

$$E_k = e_k + \frac{1}{2}|\vec{u}_k|^2 \quad , \qquad H_k = h_k + \frac{1}{2}|\vec{u}_k|^2, \qquad k = v, l,$$

where e_k is the internal energy, and $h_k = e_k + \frac{p}{\rho_k}$ the enthalpy associated to phase k and

$$\begin{aligned} \rho_m &= \alpha_g\rho_g + \alpha_l\rho_l \\ \vec{u}_m &= \frac{\alpha_g\rho_g\vec{u}_g + \alpha_l\rho_l\vec{u}_l}{\alpha_g\rho_g + \alpha_l\rho_l} \\ h_m &= \frac{\alpha_g\rho_g h_g + \alpha_l\rho_l h_l}{\alpha_g\rho_g + \alpha_l\rho_l}. \end{aligned}$$

We need a drift correlation for the relative velocity:

$$\vec{u}_r = \vec{u}_g - \vec{u}_l = \vec{f_r}(c_g, \vec{u}_m, \rho_m).$$

The phase change is modeled using the formula

$$\Gamma_g = \begin{cases} \frac{\Phi}{\mathcal{L}} & \text{if } h_l^{sat} \le h < h_g^{sat} \text{ and } 0 < \alpha_g < 1 \\ 0 & \text{otherwise} \end{cases} . \tag{4.4}$$

The isothermal two-fluid model

The model consists in the phasic mass and momentum balance equations. The main unknowns are α, P, $\vec{u}_g$, $\vec{u}_l$. The model uses stiffened gas laws $p_g(\rho_g)$ and $p_l(\rho_l)$ for a contant temperature. The subscript k stands for l the liquid phase and g the gas phase. The common averaged pressure of the two phases is denoted by p. In our model, pressure equilibrium between the two phases is postulated, and the resulting system to solve is:

$$\begin{cases} \frac{\partial m_g}{\partial t} + \nabla \cdot \vec{q}_g & = 0, \\ \frac{\partial m_l}{\partial t} + \nabla \cdot \vec{q}_l & = 0, \\ \frac{\partial \vec{q}_g}{\partial t} + \nabla \cdot (\vec{q}_g \otimes \frac{\vec{q}_g}{m_g}) + \alpha_g \vec{\nabla} p & \\ \quad + \Delta p \nabla \alpha_g - \nu_g \Delta \vec{u}_g & = m_g \vec{g} - K_g m_g ||\vec{u}_g|| \vec{u}_g \\ \frac{\partial \vec{q}_l}{\partial t} + \nabla \cdot (\vec{q}_l \otimes \frac{\vec{q}_l}{m_l}) + \alpha_l \vec{\nabla} p & \\ \quad + \Delta p \nabla \alpha_l - \nu_l \Delta \vec{u}_l & = m_l \vec{g} - K_l m_l ||\vec{u}_l|| \vec{u}_l, \end{cases} \tag{4.5}$$

where $\alpha_g + \alpha_l = 1$, $m_k = \alpha_k \rho_k$ and $\vec{q}_k = \alpha_k \rho_k \vec{u}_k$. Here, ν_k is the viscosity of phase k, and Δp denotes the default pressure $p - p_k$ between the bulk average pressure and the interfacial average pressure.

The five equation two-fluid model

The model consists in the phasic mass and momentum balance equations and one mixture total energy balance equation. The main unknowns are α,P,$\vec{u}_g$,$\vec{u}_l$ and $T = T_g = T_l$. The model uses stiffened gas laws $p_g(\rho_g, T)$ and $p_l(\rho_l, T)$.

$$\begin{cases} \frac{\partial m_g}{\partial t} + \nabla \cdot \vec{q}_g & = \Gamma_g(h_g, \Phi), \\ \frac{\partial m_l}{\partial t} + \nabla \cdot \vec{q}_l & = \Gamma_l(h_l, \Phi), \\ \frac{\partial \vec{q}_g}{\partial t} + \nabla \cdot (\vec{q}_g \otimes \frac{\vec{q}_g}{m_g}) + \alpha_g \nabla p & \\ \quad + \Delta p \nabla \alpha_g - \nu_g (\Delta \frac{\vec{q}_g}{m_g}) & = m_g \vec{g} - K_g m_g ||\vec{u}_g|| \vec{u}_g \\ \frac{\partial \vec{q}_l}{\partial t} + \nabla \cdot (\vec{q}_l \otimes \frac{\vec{q}_l}{m_l}) + \alpha_l \nabla p & \\ \quad + \Delta p \nabla \alpha_l - \nu_l (\Delta \frac{\vec{q}_l}{m_l}) & = m_l \vec{g} - K_l m_l ||\vec{u}_l|| \vec{u}_l, \\ \partial_t \rho_m E_m + \nabla \cdot (\alpha_g \rho_g H_g{}^t \vec{u}_g + \alpha_l \rho_l H_l{}^t \vec{u}_l) & = \Phi + \rho \vec{g} \cdot \vec{u} - K_g m_g ||\vec{u}_g||^3 - K_l m_l ||\vec{u}_l||^3 \end{cases}$$

where

$$\begin{aligned} \rho_m &= \alpha_g\rho_g + \alpha_l\rho_l \\ E_m &= \frac{\alpha_g\rho_g E_g + \alpha_l\rho_l E_l}{\alpha_g\rho_g + \alpha_l\rho_l}. \end{aligned}$$

The phase change is modeled using the formula

$$\Gamma_g = \begin{cases} \frac{\Phi}{\mathcal{L}} & \text{if } h_l^{sat} \leq h < h_g^{sat} \text{ and } 0 < \alpha_g < 1 \\ 0 & \text{otherwise} \end{cases}. \tag{4.6}$$

4.5 The numerical methods

CoreFlows gives a variety of finite volume methods (see [25] for an introduction). The method can be explicit or implicit, upwind or centered as in [24]. The basic method for non scalar fluid model is the Roe scheme [21] with entropic correction [23] and/or source upwinding [22].

The finite volume discretization allows an easy handling of general geometries and meshes generated by Salomé[26].
Explicit schemes are used in general for fast dynamics solved with small time steps while implicit schemes allow the use of large time steps to quickly reach the stationary regime. The implicit schemes result in nonlinear systems that are solved using a Newton type method.
The upwind scheme is the basic scheme but options are available to use a centered scheme (second order in space) or entropic corrections.

The four numerical methods available in CoreFlows to discretize the drift model are :

- Upwind scheme,
- Centered scheme,
- Low-Mach scheme,
- Staggered scheme.

more details on these methods and schemes are given in sub-section 3.2.

4.6 IT development

CoreFlows was developed in the object-oriented programming language: C++, using standard libraries and multi-platform stl.
CoreFlows relies on the toolbox [9] of the project CDMATH [8] for the handling of meshes and fields, and on the library Petsc [31] (version 3.4.5) for the handling of large sparse matrices.

4.6.1 Software architecture

CoreFlows is composed of 6 concrete classes dealing with specific models. They are listed in chronological order

- **SinglePhase** implementing the compressible Navier-Stokes equations (section 4.3)
- **DriftModel** implementing the 4 equation drift model (section 4.4)
- **IsothermalTwoFluid** implementing the isentropic two-fluid model (section 4.4)
- **FiveEqsTwoFluid** implementing the equal temperature two fluid model (section 4.4)
- **TransportEquation** implementing a scalar advection equation for the fluid enthalpy (section 4.2.1)
- **DiffusionEquation** implementing a scalar heat equation for the Uranium rods temperature (section 4.2.1)

On top of these classes there are two abstract classes that mutualize functions that are common to several models.

- ProblemFluid which contains the methods that are common to the non scalar models : SinglePhase DriftModel IsothermalTwoFluid and FiveEqsTwoFluid
- ProblemCoreFlows which contains the methods that are common to the scalar and non scalar models: ProblemFluid, TransportEquation and DiffusionEquation

Here follows 4.1 an inheritance diagram of CoreFlows

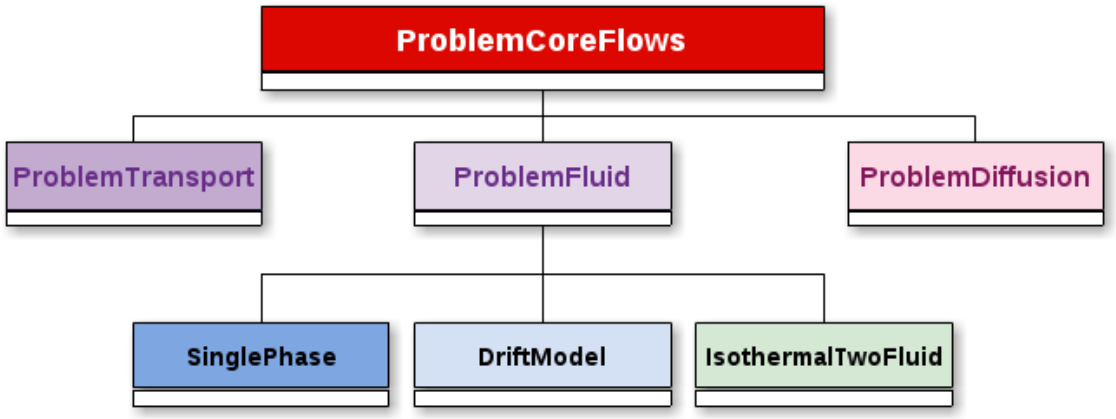

Figure 4.1: Inheritance diagram of CoreFlows

The program can build simple geometries and meshes using the library CDMATH [8] or read complex geometries and meshes written with the MED file system (see [26]). The output files containing the fields resulting from the calculation can be either of VTK ([28]) or MED type. One can use Paraview [27] or Salome [26] to visualise the results.
Vector and matrices structures come from the Petsc [31] library. The matrices are stored in a block sparse format (type baij in Petsc conventions). The default linear solver is GMRES and the default preconditioner is ILU, both provided by Petsc.

4.6.2 Software engineering

CMake

CoreFlows uses the CMake production engine.
CMake is cross-platform free and open-source software for managing the build process of software using a compiler-independent method. It is designed to support directory hierarchies and applications that depend on multiple libraries. It is used in conjunction with native build environments such as make. It has minimal dependencies, requiring only a C++ compiler on its own build system

CMake can handle in-place and out-of-place builds, enabling several builds from the same source tree, and cross-compilation. The ability to build a directory tree outside the source tree is a key feature, ensuring that if a build directory is removed, the source files remain unaffected

CoreFlows uses also "**gcov**", which is a source code coverage analysis and statement-by-statement profiling tool. It generates exact counts of the number of times each statement in a program is executed and annotates source code to add instrumentation. The gcov utility gives information on how often a program executes segments of code.

Graphical user interface 'GUI' : Qt-Designer

Qt Designer is Qt's tool for designing and building graphical user interfaces (GUIs) from Qt [36] components. We can compose and customize our widgets or dialogs in a what-you-see-is-what-you-get *(WYSIWYG)* manner, and test them using different styles and resolutions. Widgets and forms created with Qt Designer integrated seamlessly with programmed code, using Qt's signals and slots mechanism, that lets us easily assign behavior to graphical elements. All properties set in Qt Designer can be changed dynamically within the code.

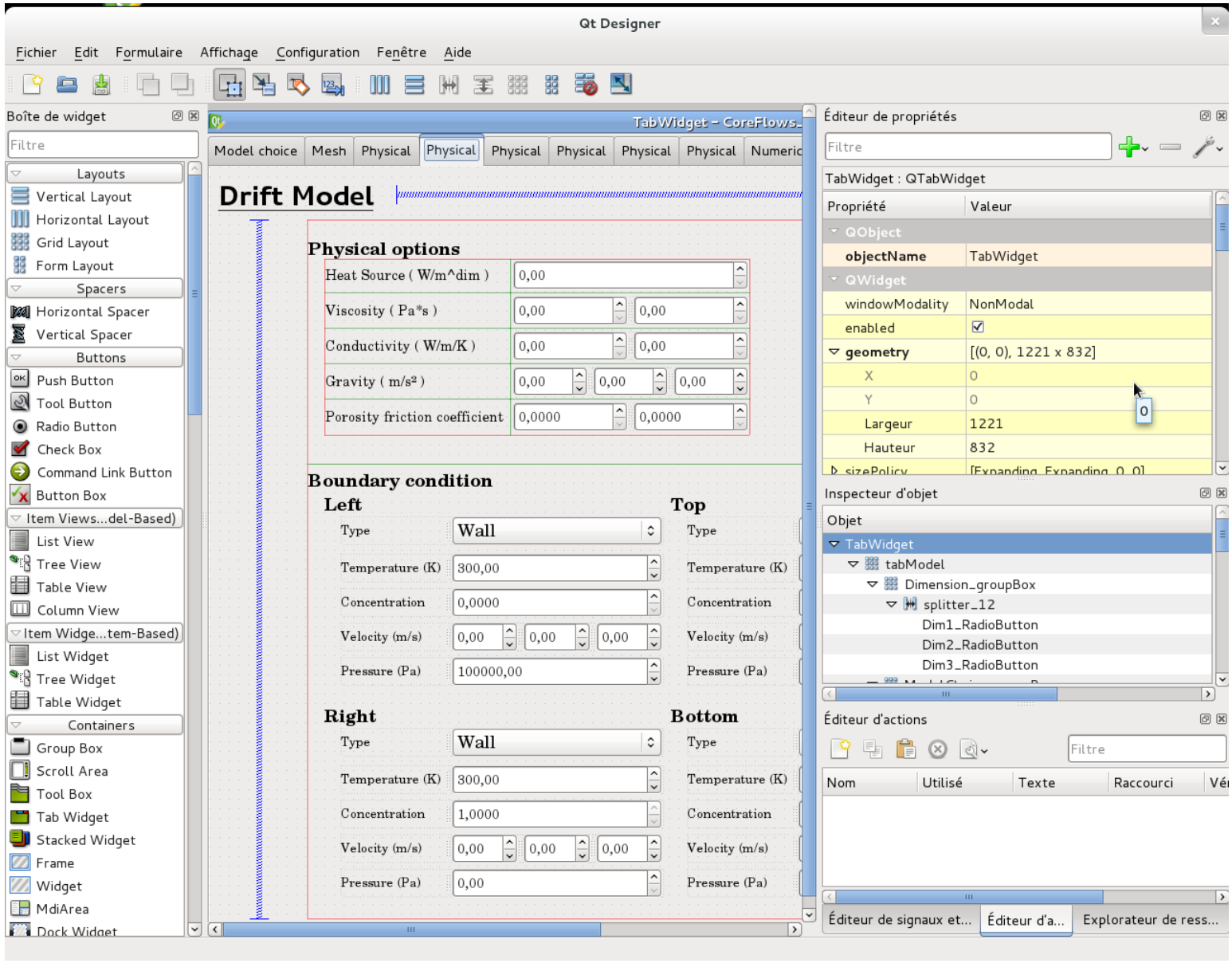

Figure 4.2: Development environment CoreFlows's GUI

API *Python* : SWIG

SWIG[11] is an open source software tool used to connect computer programs or libraries written in C or C++ with scripting languages such as *Python*.

There are two main reasons to embed a scripting engine in an existing C/C++ program:

- The program can then be customized far faster, via a scripting language instead of C/C++.
- Even if the final product is not to contain the scripting engine, it may nevertheless be very useful for writing test scripts.

SWIG's generating mechanism is shown in figure 4.3.

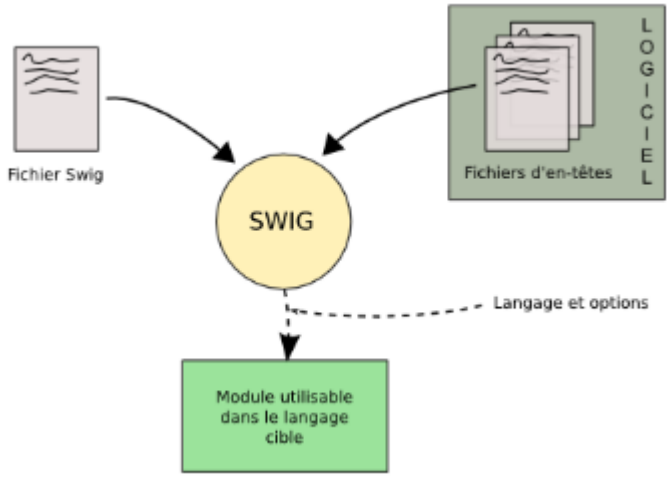

Figure 4.3: SWIG's generating mechanism

The result is a *Python* module, usable by a person knowing CoreFlows methods. *Python*'s API is favored by the flexibility of its language, its interactivity and its readable prototyping. The CoreFlows's documentation remains valid in *Python*.

The SWIG has brought the power of *Python* to the performance of C ++.

11. Simplified Wrapper and Interface Generator

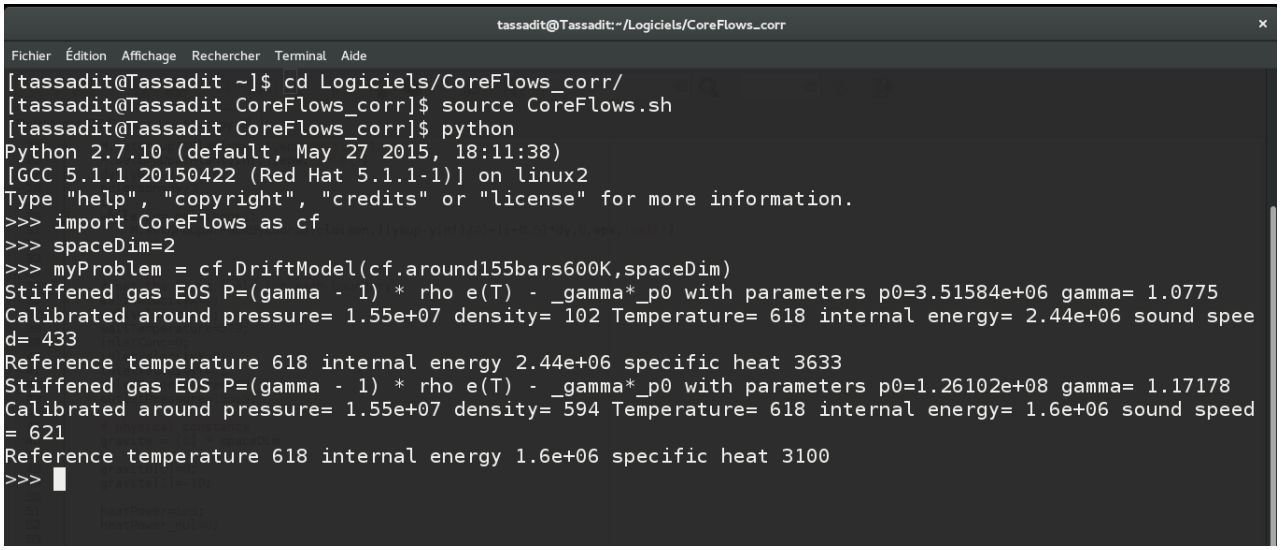

Figure 4.4: Example of CoreFlows's use on *Python*

4.6.3 Documentation : Doxygen

Doxygen is a documentation generator, a tool for writing software reference documentation. The documentation is written within code, and is thus relatively easy to keep up to date. Doxygen can cross reference documentation and code, so that the reader of a document can easily refer to the actual code.
Doxygen supports multiple programming languages, in particular C++, C, C#, Python, ...
Doxygen allows one to extract the following information contained in the source code :

+ Prototype and documentation of classes and their hierarchy;
+ Different types of graphs: Class diagrams, ...
+ Documentation of data structures;
+ List of included files;
+ Prototype and documentation of functions, whether local, private or public, etc. ;
+ List of modules ;
+ An index of all identifiers;
+ Source files annotated (for example with line numbers) and Navigable (eg with HTML, with which the identifiers refer to the associated documentation).

CoreFlows 1.0

A library providing basic two phase flow models and finite volume numerical methods for students and researchers in computational multiphase fluid dynamics

Main Page | Related Pages | Classes | Files | Search

CoreFlows User Guide

Presentation of CoreFlows

CoreFlows is an open source C++/Python library intended at solving PDE systems arising from the thermalhydraulics of two phase flows in power plant boilers. It is a simple environment meant at students and researchers to test new numerical methods on general geometries with unstructured meshes. It is developped at CEA Saclay by Michael Ndjinga and his students since 2014 and proposes a few basic models and finite volume numerical methods. Some of the main objectives are the study of

- Numerical schemes for compressible flows at low Mach numbers
- Well balanced schemes for stiff source terms (heat source, phase change, pressure losses)
- Flow inversion and counter-current two phase flows
- Schemes that preserve the phasic volume fraction $\alpha \in [0, 1]$
- Convergence of finite volume methods
- New preconditioners for implicit methods for two phase flows
- The coupling of fluid models or multiphysics coupling (eg thermal hydraulics and neutronics or thermal hydraulics and solid thermics)

CoreFlows relies on the **Toolbox [21]** of the project **CDMATH [20]** for the handling of meshes and fields, and on the library **Petsc [19]** 3.4.5 for the handling of large sparse matrices.

Contents

This document is the user guide of the CoreFlows library. It is organized as follows :

- **The Physical Models**
  - **The linear scalar problems**
    1. **The transport equation**
    2. **The diffusion equation**
  - **The Navier-Stokes model**
  - **The two-phase flow models**
    1. **The Drift model**
    2. **The isothermal two-fluid model**
    3. **The five equation two-fluid model**
- **Software structure**
- **The numerical methods**
- Summary of available **functionalities**
- **CoreFlows example scripts**

Installation and use

In order to install CoreFlows you will need the packages **CMAKE [25]** , **HDF5 [23]** and possibly

- **SWIG [22]** if you want to use python scripts
- **DOXYGEN [24]** if you want to generate this html documentation.

Instructions for installation and use of CoreFlows can be found here **Installation and use of CoreFlows**.

References

A specific page dedicated to references is available here.

Generated on Tue Sep 1 2015 11:56:14 for CoreFlows by doxygen 1.8.9.1

Figure 4.5: CoreFlows's home page

4.7 Examples of CoreFlows's use

4.7.1 Graphical user interface 'GUI'

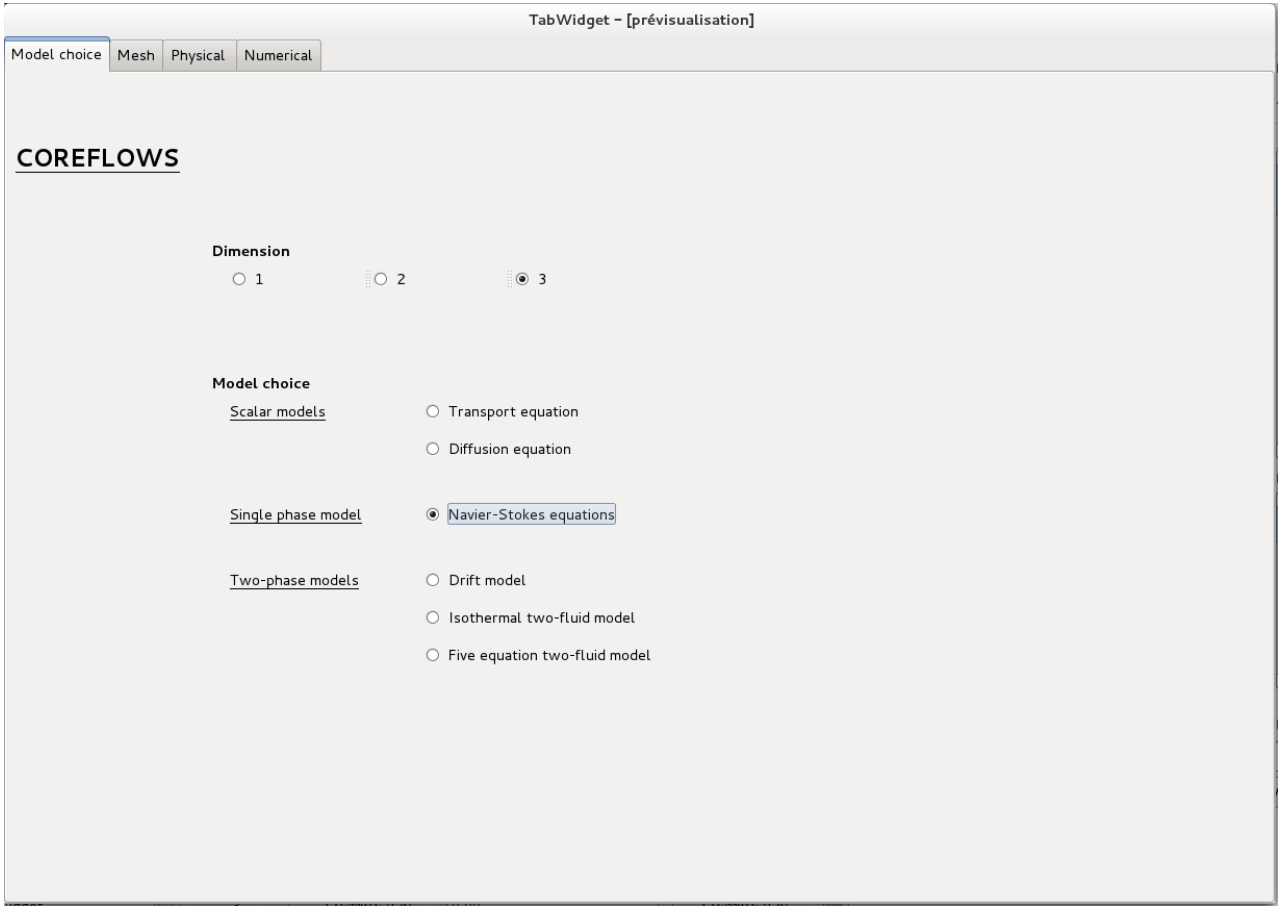

Figure 4.6: CoreFlows's GUI 1/4

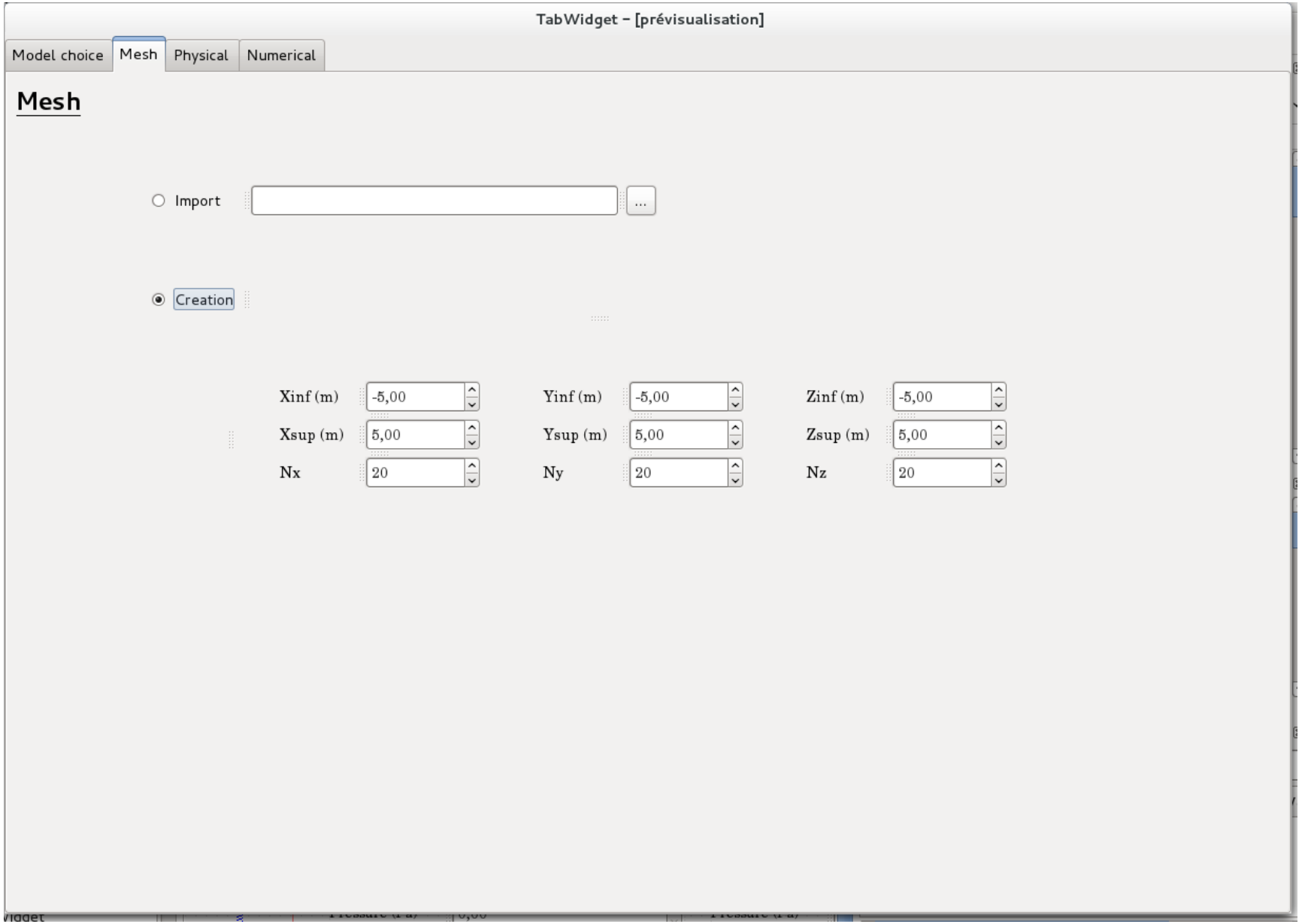

Figure 4.7: CoreFlows's GUI 2/4

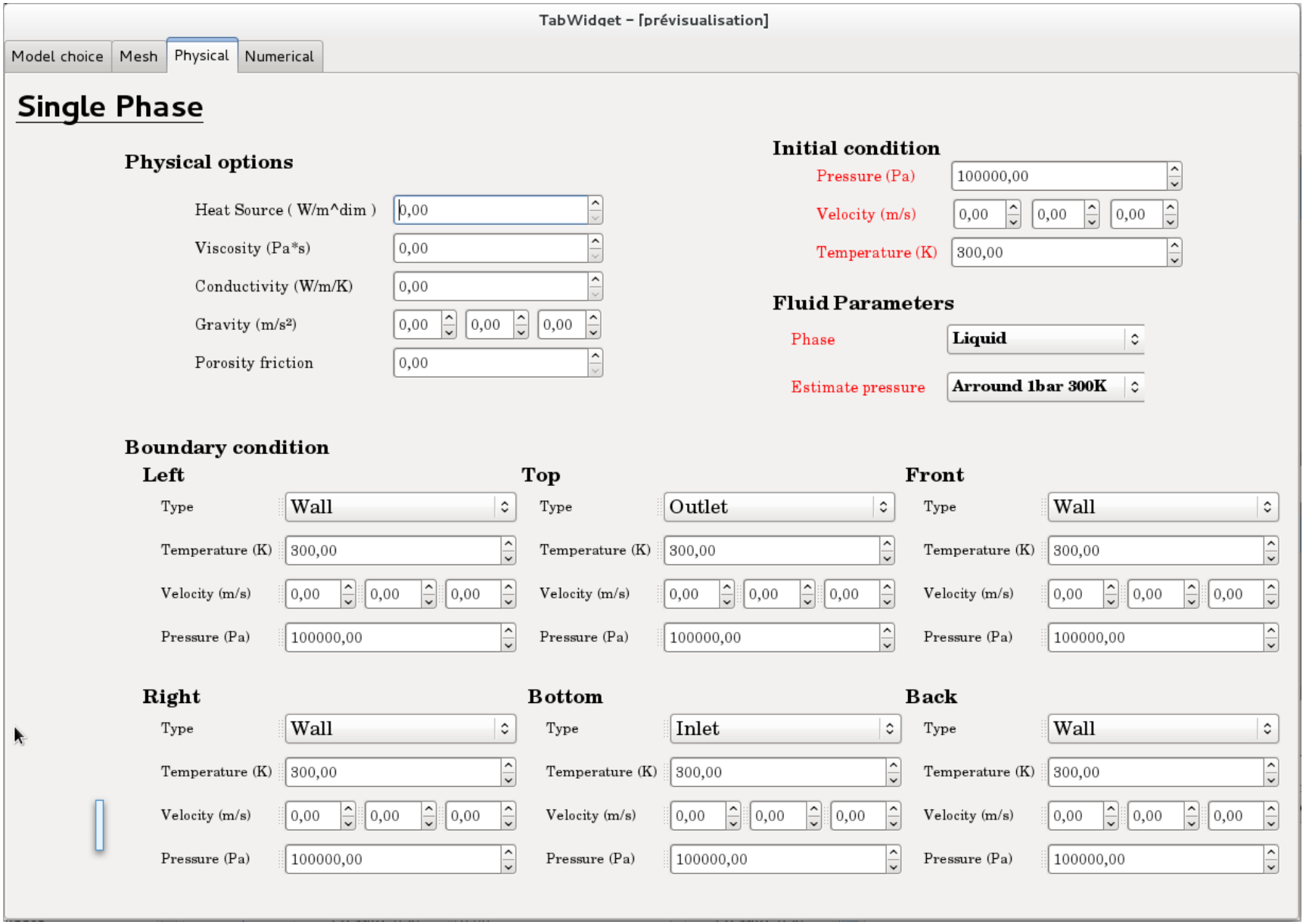

Figure 4.8: CoreFlows's GUI 3/4

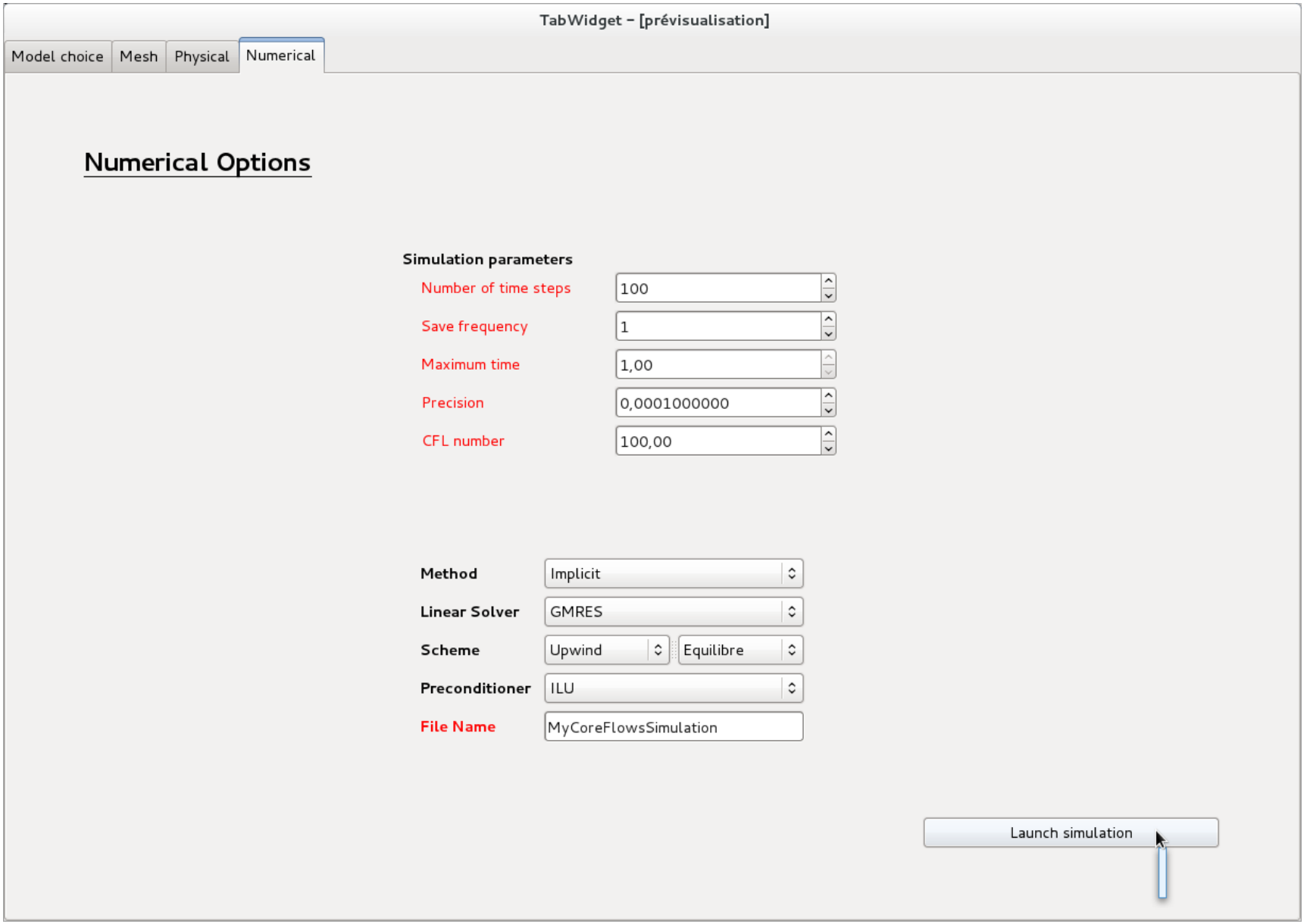

Figure 4.9: CoreFlows's GUI 4/4

4.7.2 Scripts *Python*

Although CoreFlows's GUI is very simple to use, we cannot make too complex calculations, therefore the use of *Python* scripts may be interesting in that case.

Example 1 : Transport Equation

```
 import CoreFlows as cf

def TransportEquation_1DHeatedChannel():
spaceDim = 1;
    # Prepare for the mesh
xinf = 0 ;
xsup=4.2;
nx=2;
    # set the limit field for each boundary
inletEnthalpy=1.3e6;
    # Set the transport velocity
transportVelocity=[5];
myProblem = cf.TransportEquation(cf.Liquid,cf.around155bars600K,
transportVelocity);
nVar = myProblem.getNumberOfVariables();
    # Prepare for the initial condition
VV_Constant = [1.3e6]; #initial enthalpy
#Set rod temperature and heat exchamge coefficient
rodTemp=623;#Rod clad temperature
heatTransfertCoeff=1000;#fluid/solid heat exchange coefficient
myProblem.setRodTemperature(rodTemp);
myProblem.setHeatTransfertCoeff(heatTransfertCoeff);
    #Initial field creation
print("Building initial data " );
myProblem.setInitialFieldConstant(spaceDim,VV_Constant,xinf,xsup,nx,"inlet",
"neumann");
    # Set the boundary conditions
myProblem.setInletBoundaryCondition("inlet", inletEnthalpy);
myProblem.setNeumannBoundaryCondition("neumann")
    # Set the numerical method
myProblem.setNumericalMethod(cf.upwind, cf.Explicit);
    # name file save
fileName = "1DHeatedChannel";
```

```
    # parameters calculation
MaxNbOfTimeStep = 3 ;
freqSave = 5;
cfl = 0.95;
maxTime = 5;
precision = 1e-6;
myProblem.setCFL(cfl);
myProblem.setPrecision(precision);
myProblem.setMaxNbOfTimeStep(MaxNbOfTimeStep);
myProblem.setTimeMax(maxTime);
myProblem.setFreqSave(freqSave);
myProblem.setFileName(fileName);
myProblem.setDISPLAY( True,True, True, True);
     # evolution
myProblem.initialize();
print("Running python "+ fileName );
ok = myProblem.run();
if (ok):
print( "Simulation python " + fileName + " is successful !" );
pass
else:
print( "Simulation python " + fileName + "  failed ! " );
pass
print( "------------ End of calculation !!! -----------" );
myProblem.terminate();
return ok
```

Example 2 : Navier–Stokes equations - single phase -

```
 import CoreFlows as cf
def SinglePhase_3DHeatDrivenCavity():
spaceDim = 3;
    #Preprocessing: mesh data
xinf=0;
xsup=1;
yinf=0;
ysup=1;
zinf=0;
zsup=1;
nx=10;
ny=10;
nz=10;
    # set the limit field for each boundary
coldWallVelocityX=0;
coldWallVelocityY=0;
coldWallVelocityZ=0;
coldWallTemperature=563;
hotWallVelocityX=0;
hotWallVelocityY=0;
hotWallVelocityZ=0;
hotWallTemperature=613;
    # physical constants
gravite = [0] * spaceDim
gravite[2]=-10;
gravite[1]=0;
gravite[0]=0;
viscosite=[8.85e-5];
conductivite=[1000];#Wall heat transfert due to nucleate boiling.
   #-------------------------------------------------------------------------
myProblem = cf.SinglePhase(cf.Liquid,cf.around155bars600K,spaceDim);
nVar = myProblem.getNumberOfVariables();
#Initial field creation
print("Building initial data " );
    # Prepare for the initial condition
VV_Constant = [0] * nVar
```

```
# constant vector
VV_Constant[0] = 155e5;
VV_Constant[1] = 0 ;
VV_Constant[2] = 0;
VV_Constant[3] = 0;
VV_Constant[4] = 573;
    #Initial field creation
myProblem.setInitialFieldConstant(spaceDim,VV_Constant,xinf,xsup,nx,"hotWall",
"hotWall",yinf,ysup,ny,"hotWall","hotWall",zinf,zsup,nz, "hotWall", "coldWall");
    # Set the boundary conditions
myProblem.setWallBoundaryCondition("coldWall", coldWallTemperature, coldWallVelocit
coldWallVelocityY, coldWallVelocityZ);
myProblem.setWallBoundaryCondition("hotWall", hotWallTemperature, hotWallVelocityX,
hotWallVelocityY, hotWallVelocityZ);
    # set physical parameters
myProblem.setViscosity(viscosite);
myProblem.setConductivity(conductivite);
myProblem.setGravity(gravite);
    # set the numerical method
myProblem.setNumericalMethod(cf.upwind, cf.Implicit);
myProblem.setLinearSolver(cf.GMRES,cf.ILU,True);
myProblem.setEntropicCorrection(False);
myProblem.setWellBalancedCorrection(False);
    # name file save
fileName = "3DHeatDrivenCavity";
    # simulation parameters
MaxNbOfTimeStep = 3 ;
freqSave = 1;
cfl = 10;
maxTime = 50;
precision = 1e-6;
myProblem.setCFL(cfl);
myProblem.setPrecision(precision);
myProblem.setMaxNbOfTimeStep(MaxNbOfTimeStep);
myProblem.setTimeMax(maxTime);
myProblem.setFreqSave(freqSave);
myProblem.setFileName(fileName);
myProblem.setNewtonSolver(precision,20);
myProblem.saveConservativeField(True);
```

```
if(spaceDim>1):
myProblem.saveVelocity();
pass
    # evolution
myProblem.initialize();
print("Running python "+ fileName );
ok = myProblem.run();
if (ok):
print( "Simulation python " + fileName + " is successful !" );
pass
else:
print( "Simulation python " + fileName + "  failed ! " );
pass
print( "------------ End of calculation !!! -----------" );
myProblem.terminate();
return ok
```

Chapter 5

Numerical Results

In the current section we are first going to validate the three numerical methods: Low-Mach, Upwind and Staggered, on the classical configuration of the cavity driven by its lid (in sub-section 5.1).
And then study a specific configuration called the thermal hydraulics of the dynamics of a flow controlled by a thermal conduction (sub-section 5.2).
Finally, we are going to study a specific configuration for nuclear reactors, that is to say the flow between two different channels, which is the main objective of this internship (sub-section 5.3) .

5.1 The Driven Cavity

The first investigation was to demonstrate the upwind scheme's limitations when the flow is at low-Mach number and Froude, and the dynamic almost incompressible.
We consider a 2D $[0,1]$x$[0,1]$ domain, where all the domain's edges are (fixed) walls, except the top one, which has a velocity in the $\vec{x}$ direction $U_x = 1m/s$.

We apply a $273°C$ temperature, a pressure of $155bars$ and a viscosity of 0.025. We consider the following meshes:

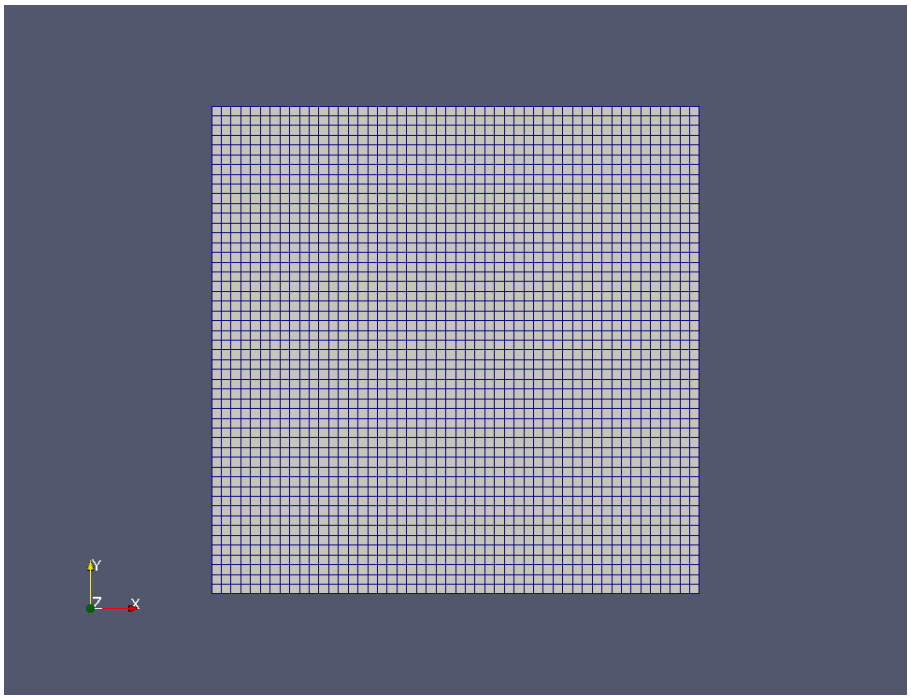

Figure 5.1: Driven Cavity : structed mesh

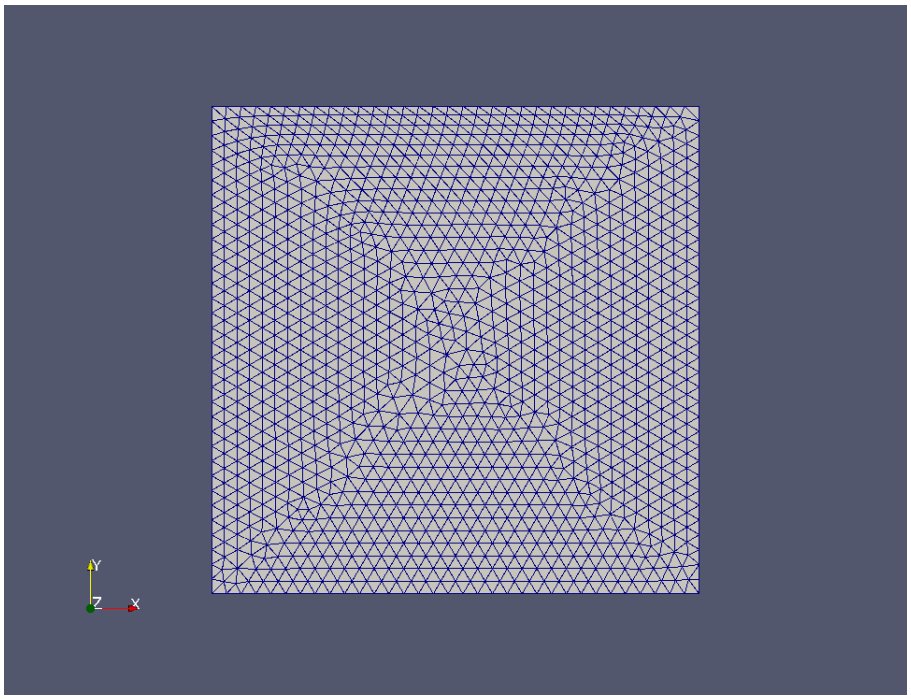

Figure 5.2: Driven Cavity : unstructed mesh

Indeed, the results of the upwind scheme are not very accurate (see 5.3 & 5.4)

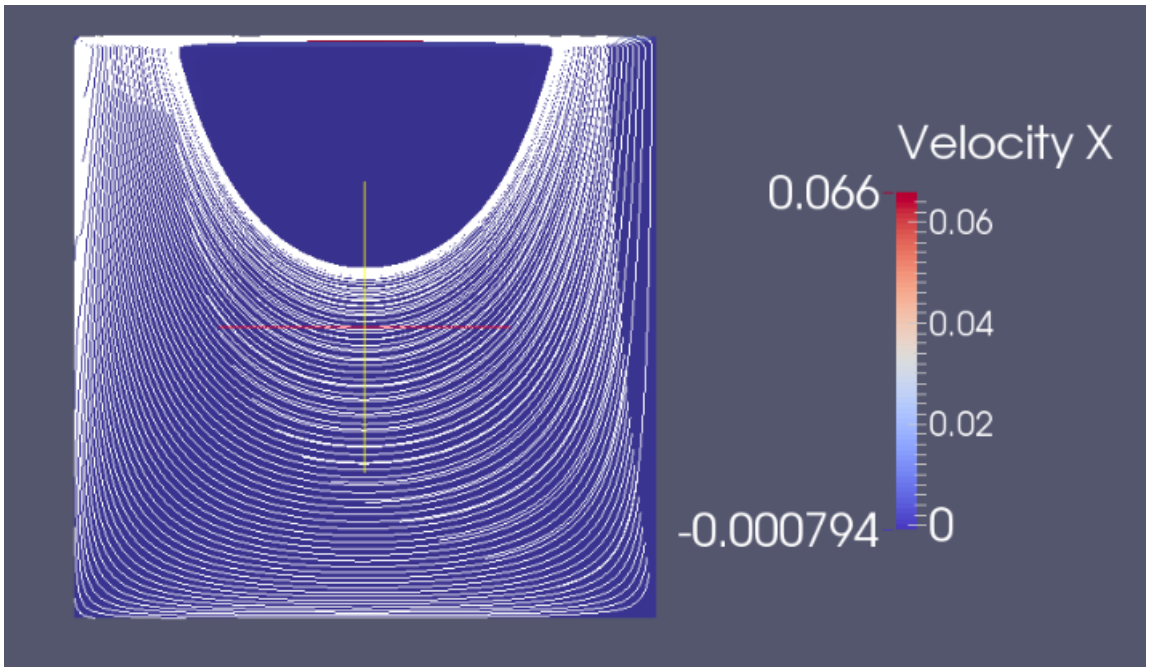

Figure 5.3: Driven Cavity : Upwind scheme - structed mesh

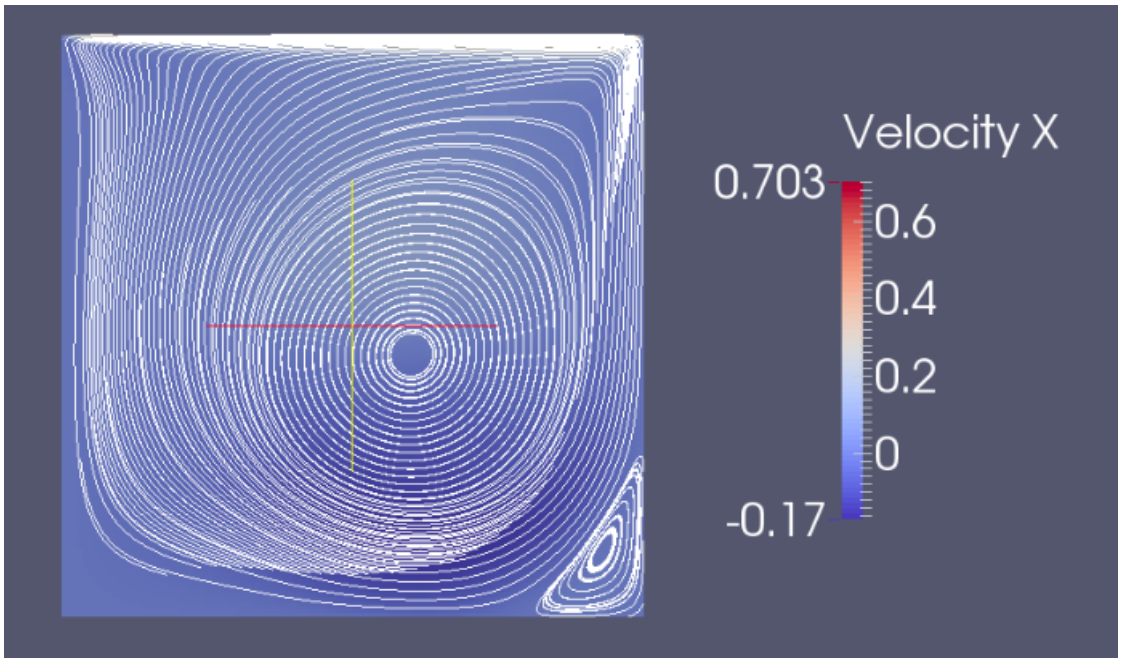

Figure 5.4: Driven Cavity : Upwind scheme - unstructed mesh

The LowMach scheme is more precise but causes robustness problems

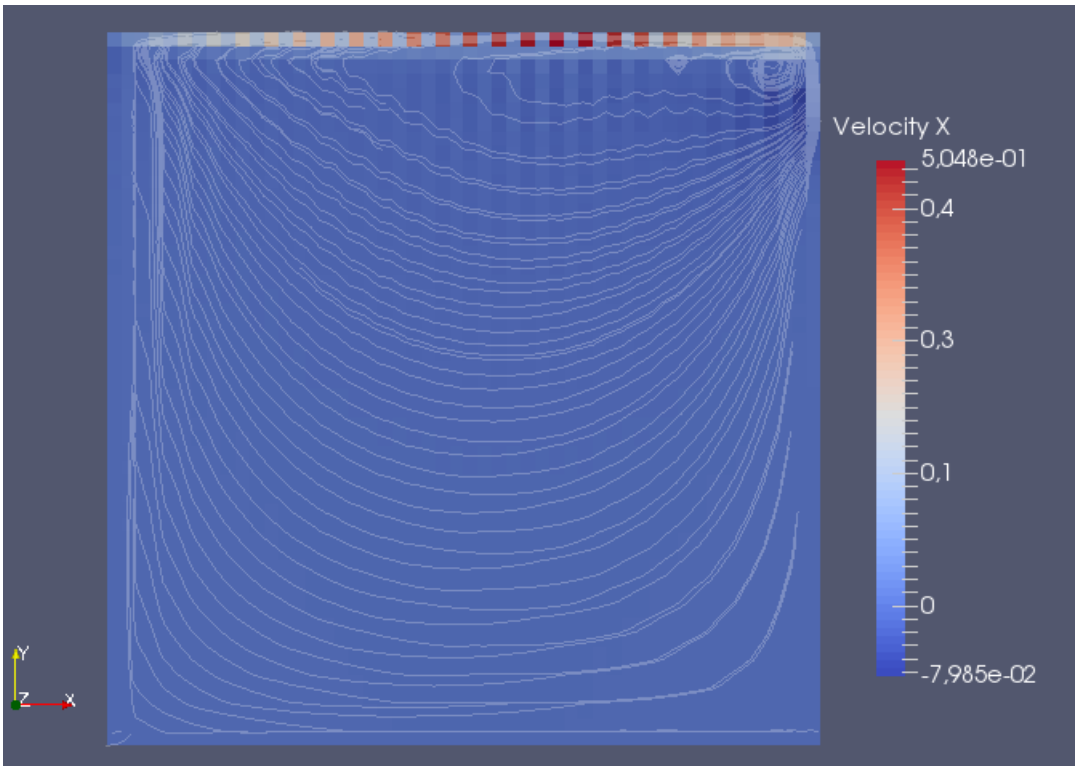

Figure 5.5: Driven Cavity : LowMach-explicite scheme - structed mesh

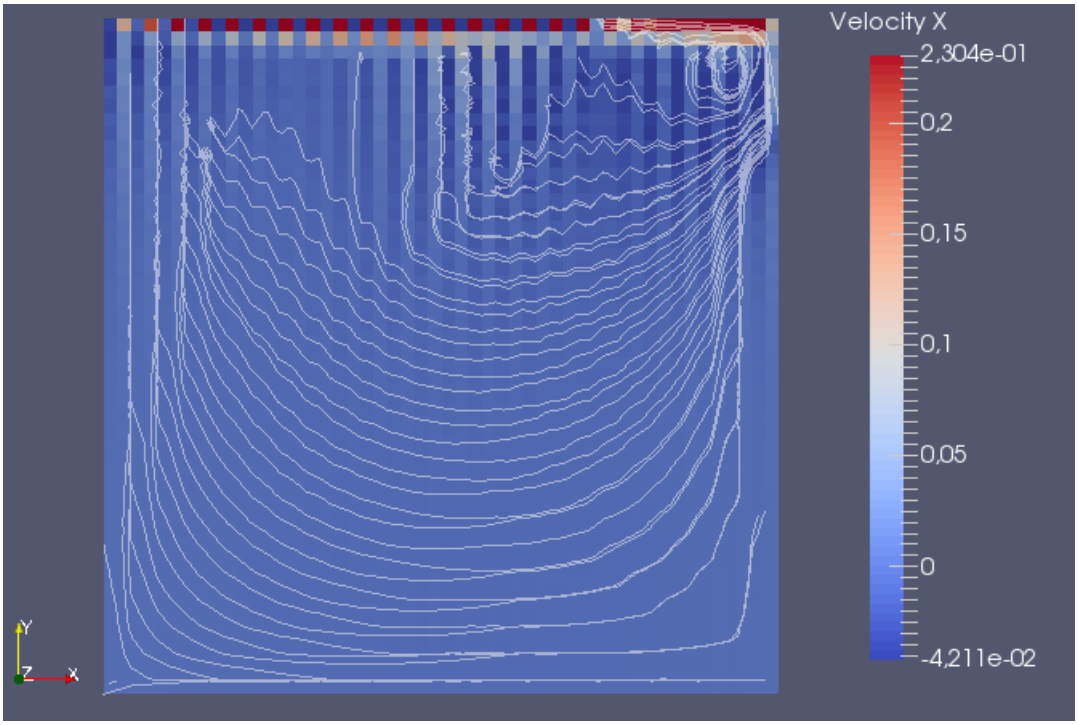

Figure 5.6: Driven Cavity : LowMach-implicite scheme - structed mesh

The staggered scheme gives accurate results without any parasitic oscillations (5.7 & 5.8).

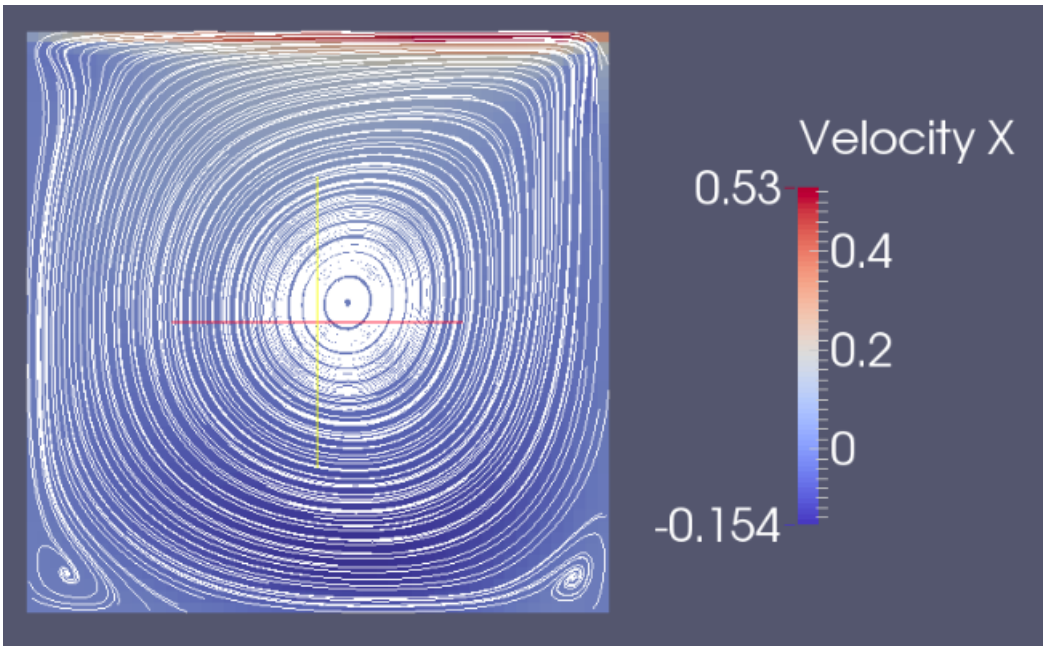

Figure 5.7: Driven Cavity : Staggered scheme - structed mesh

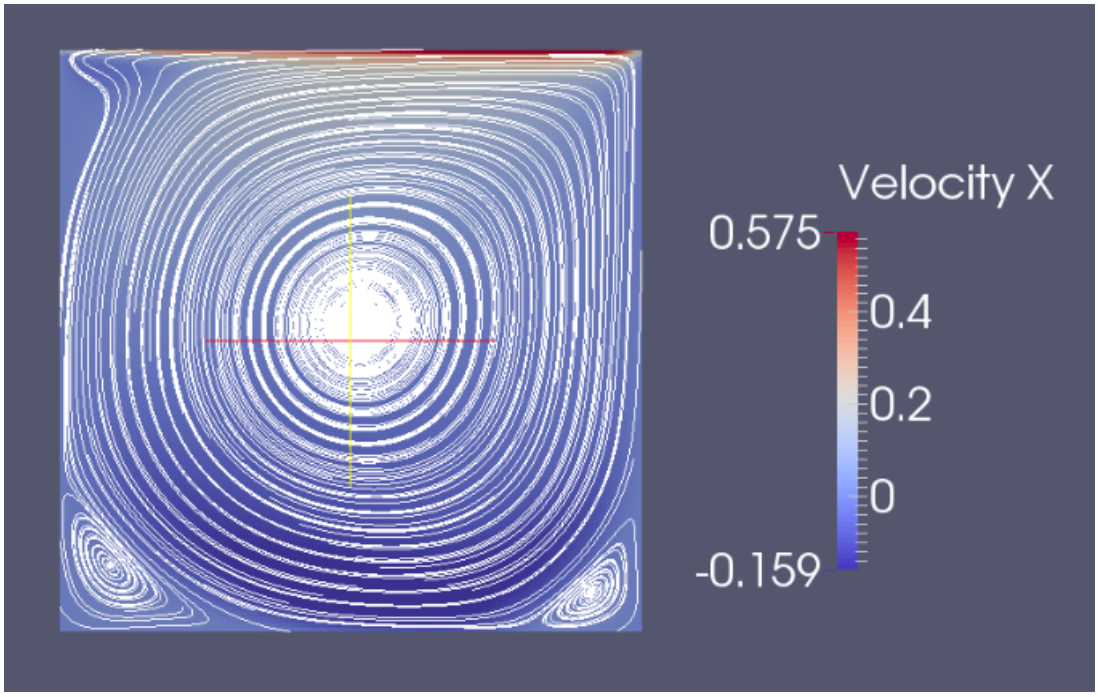

Figure 5.8: Driven Cavity : Staggered scheme - unstructed mesh

5.2 Flow led by the conduction

The second investigation is related to the Riemann problem which is a mixture of two water slides of different temperature, for instance from two channels differently heated with

The second investigation is related to the Riemann problem which is a mixture of two water slides of different temperature, for instance from two channels differently heated with $T_1 = 563K$ and $T_2 = 623K$, in $[0, 20cm]$x$[0, 40cm]$. We impose an input velocity in the $\vec{y}$ direction equals to $U_y = 1m/s$, and output pressure of $155bar$.
We consider viscosity of $1.5Pa.s$, and a conductivity of $5000W/m/s$

Figure 5.9 represents the initial configuration of the calculation with a mesh of $40x80$ elements.

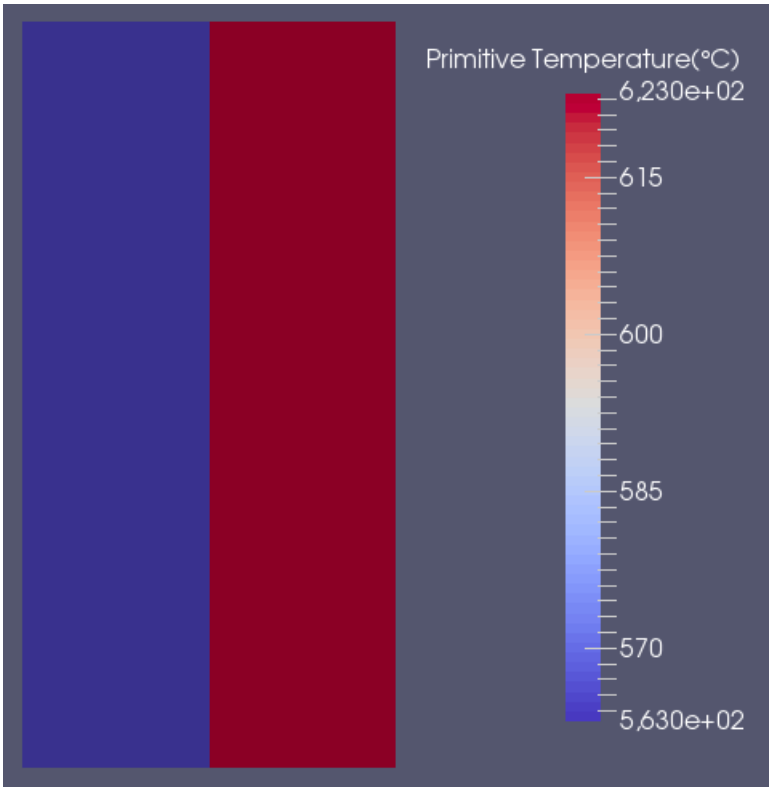

Figure 5.9: Riemann problem: $T_1 = 563K$ et $T_2 = 623K$ à $t = 0s$

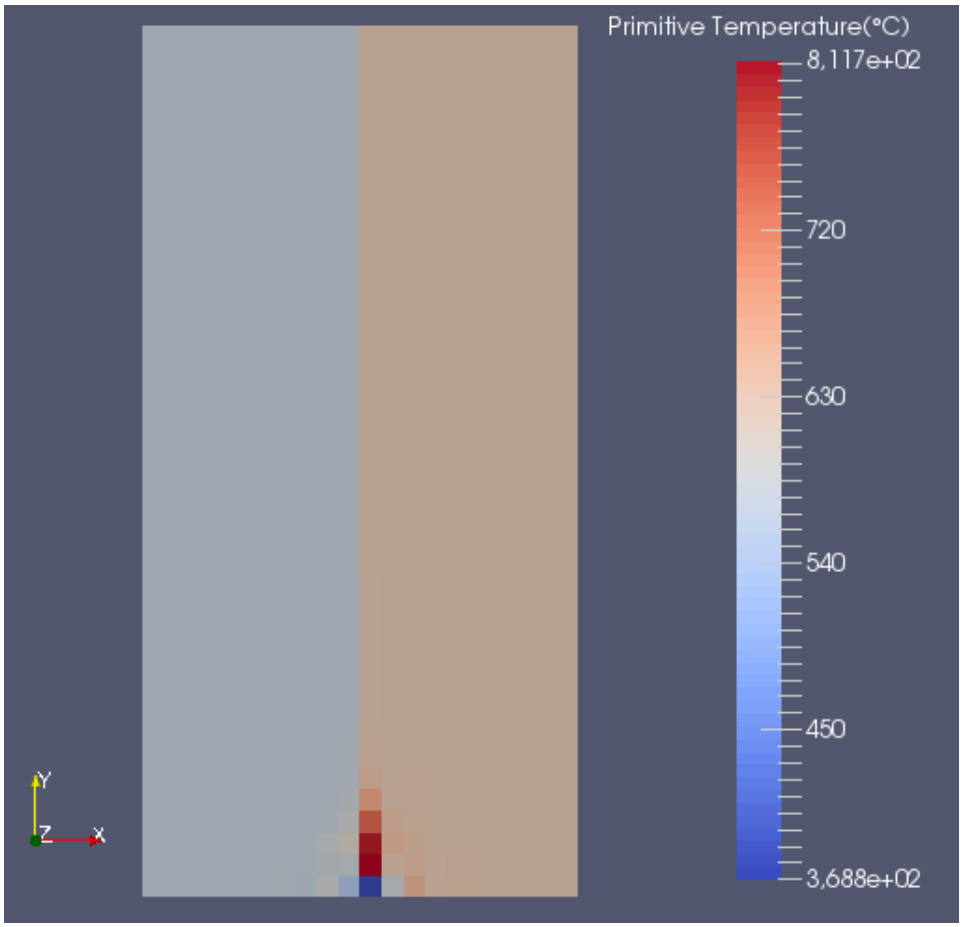

Figure 5.10: Riemann problem: LowMach scheme

As in the previous case, we notice that the low-mach scheme presents strong oscillations. The upwind and staggered schemes give similar results.
We notice that in this pure thermal calculation without any term source, the staggered scheme (figure 5.11), is less diffusive than the upwind scheme (figure 5.12).

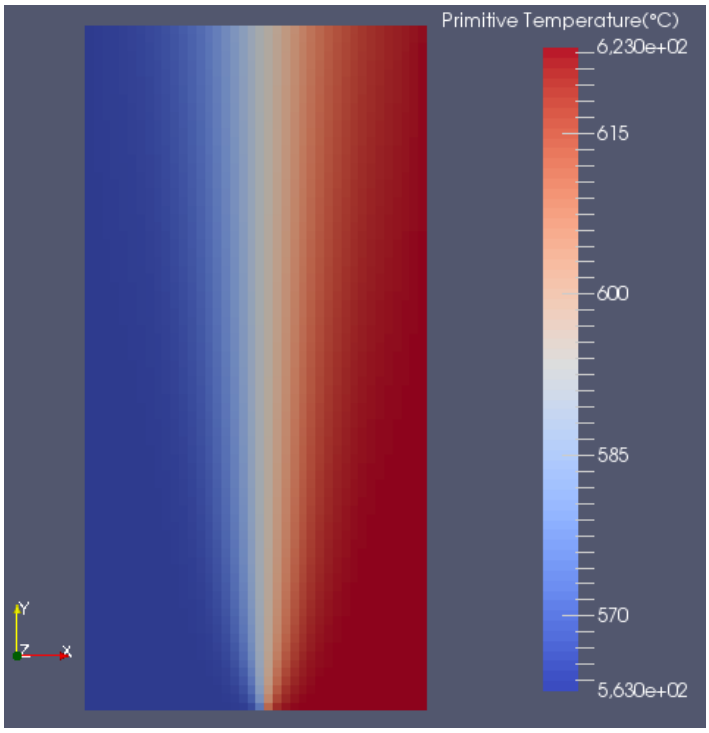

Figure 5.11: Riemann problem: Staggered scheme

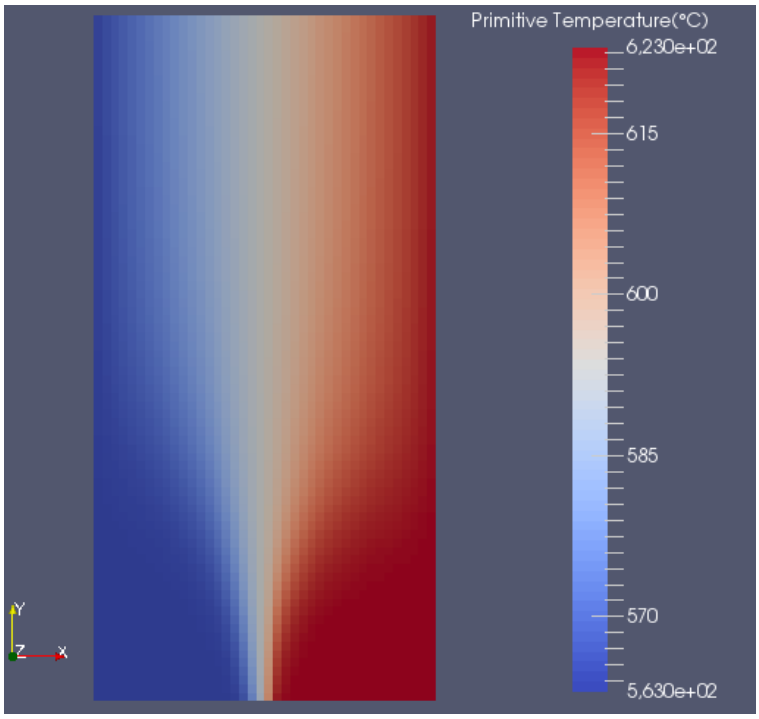

Figure 5.12: Riemann problem: Upwind scheme

5.3 Recirculation between parallel channels

The last case is the hardest, we are going to model the core of PWR900 in 2D in a rectangular domain $\Omega = [0, R] \times [0, L]$ which is heated using a flax of $10^8 W/m^2$, in $\Omega = [0, \frac{R}{2}] \times [\frac{L}{4}, \frac{3L}{4}]$ with the following values:

$$\begin{aligned} L &= 4m \\ R &= 2m. \end{aligned}$$

We impose an input velocity in the $\vec{y}$ direction $U_y = 0.75m/s$, and an output pressure of $155bar$.
We consider a gravity $\vec{g} = (0, -10)$

The figure 5.13 represents the initial configuration of the calculation of the mesh of 800 elements ($20X40$).

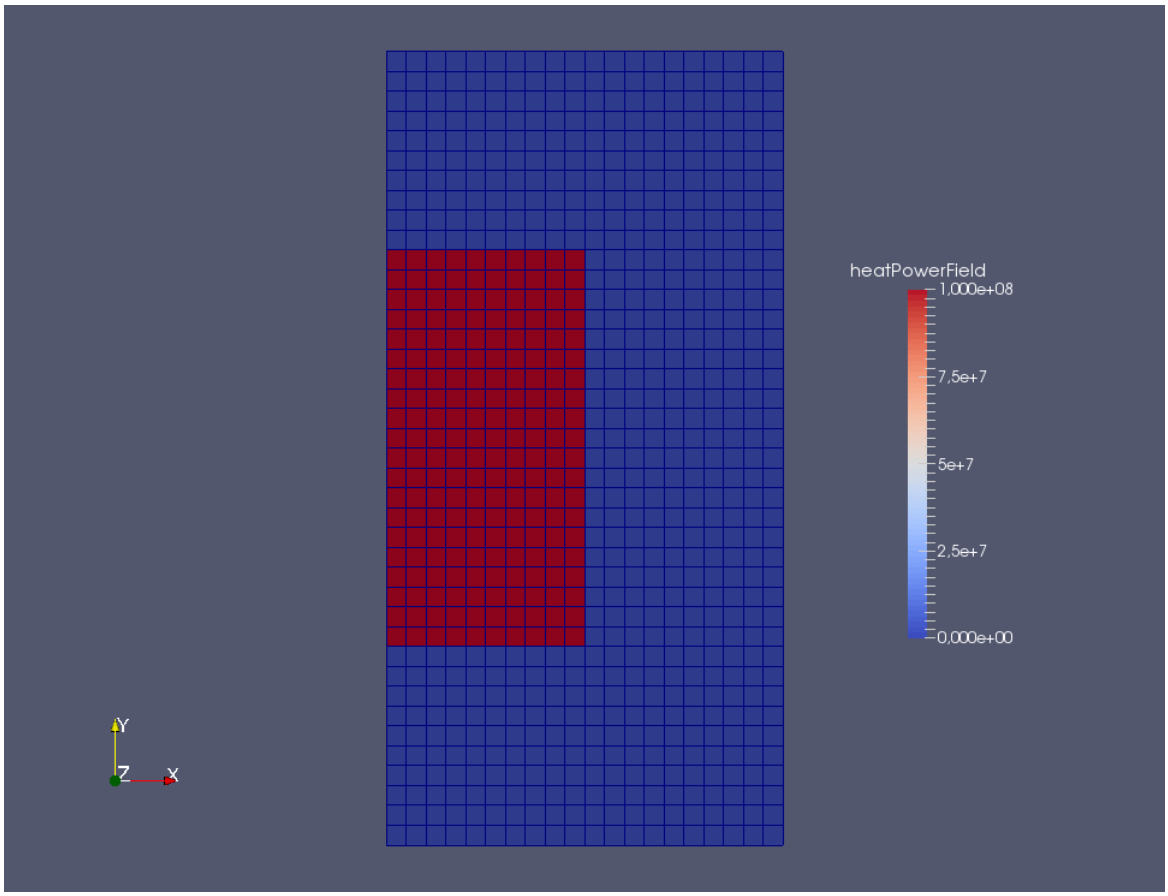

Figure 5.13: Initial configuration of channel

5.3.1 Single phase case

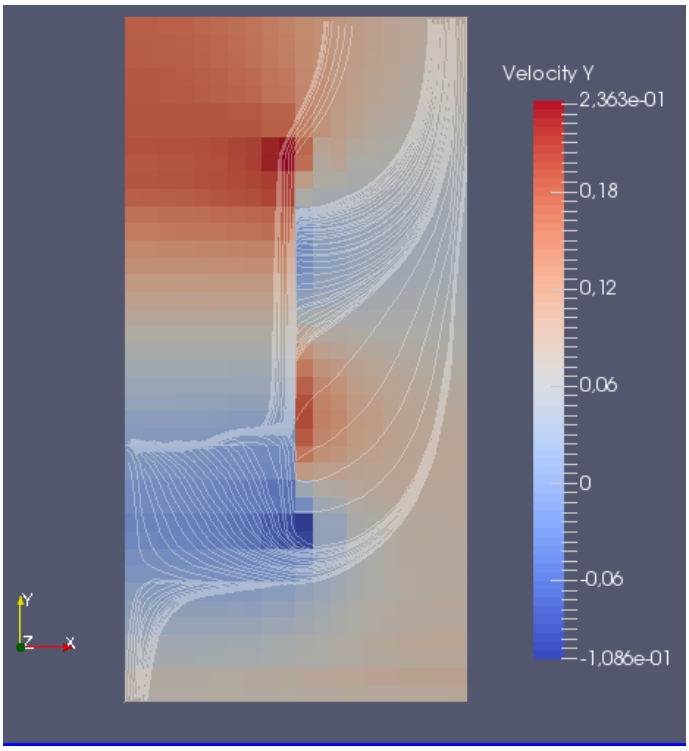

Figure 5.14: Single phase - Velocity U_y, LowMach Scheme

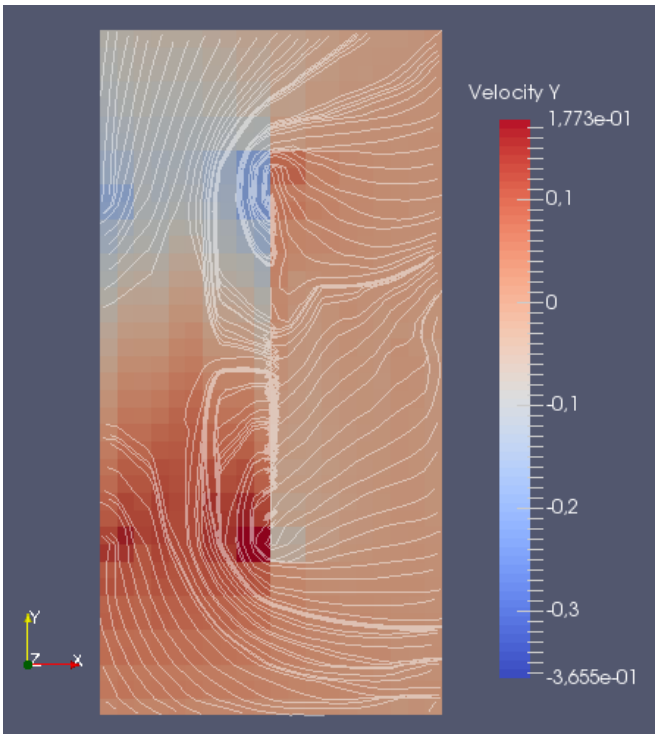

Figure 5.15: Single phase - Velocity U_y, LowMach-well-balanced Scheme

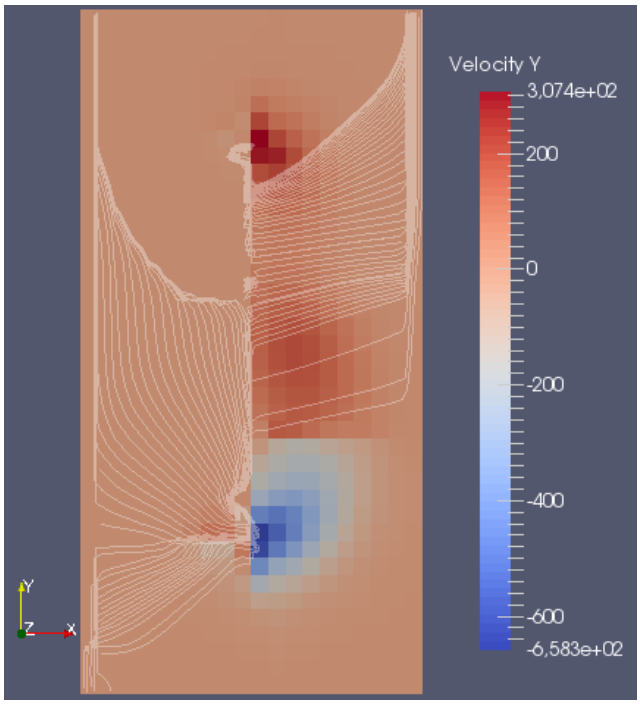

Figure 5.16: Single phase - Velocity U_y, Staggered Scheme

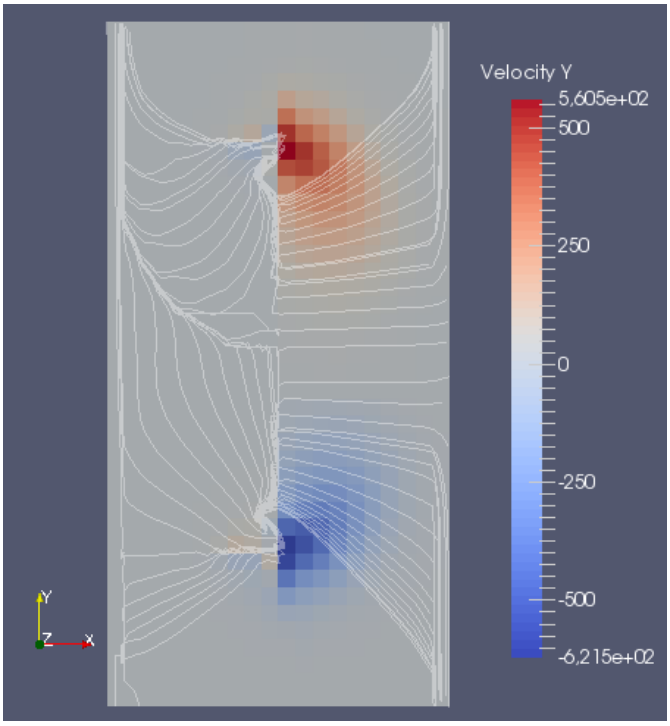

Figure 5.17: Single phase - Velocity U_y, Staggered-well-balanced Scheme

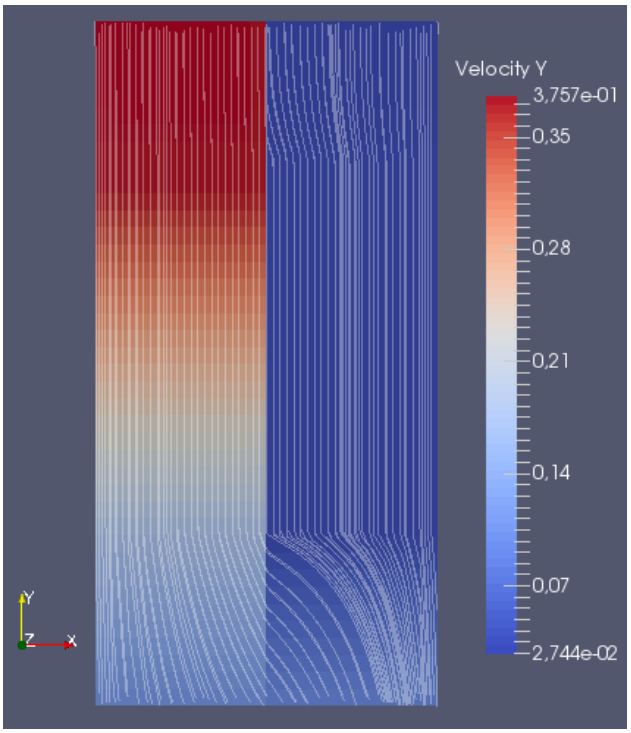

Figure 5.18: Single phase - Velocity U_y, Upwind Scheme

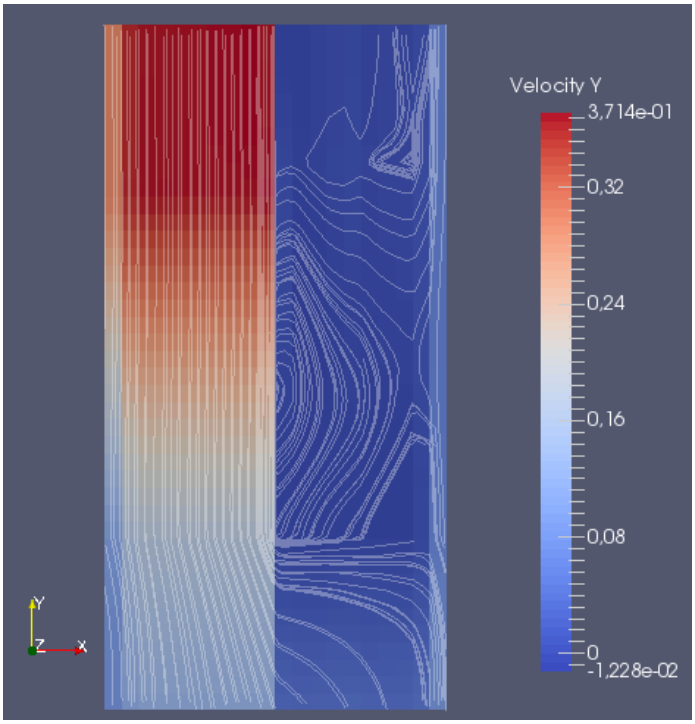

Figure 5.19: Single phase - Velocity U_y, Upwind-well-balanced Scheme

Single phase case : inclined

In this case, we take the same parameters as in the section 5.3, except the gravity. In this case, g=(7,-7). The image 5.20 represents the initial configuration of the channel.

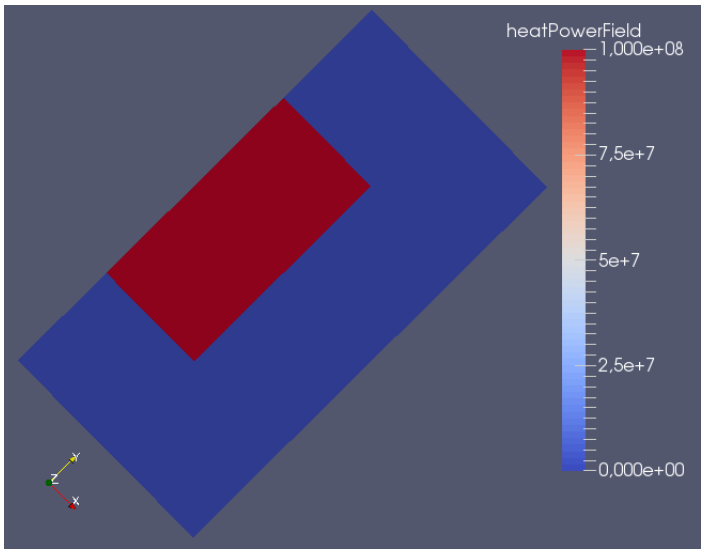

Figure 5.20: Initial configuration of inclined channel

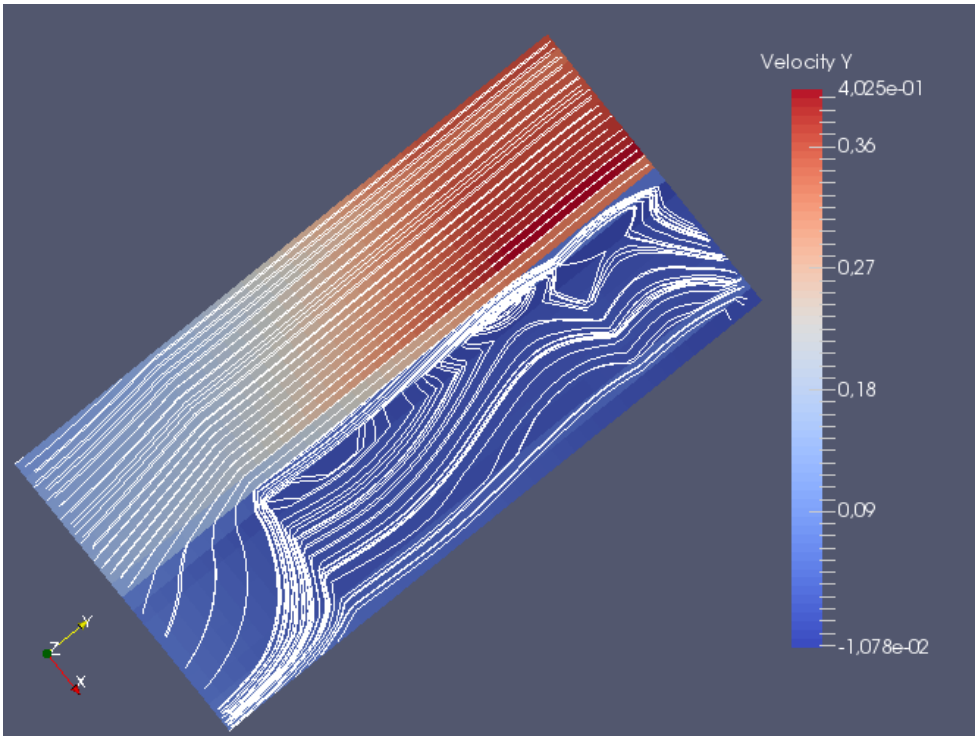

Figure 5.21: Single phase - inclined channel - Single phase - Velocity U_y, Upwind-well-balanced Scheme

The LowMach and staggered case blow up because of the presence of a wall. Only the upwind scheme goes until the end of the calculation.

5.3.2 Two-phase case

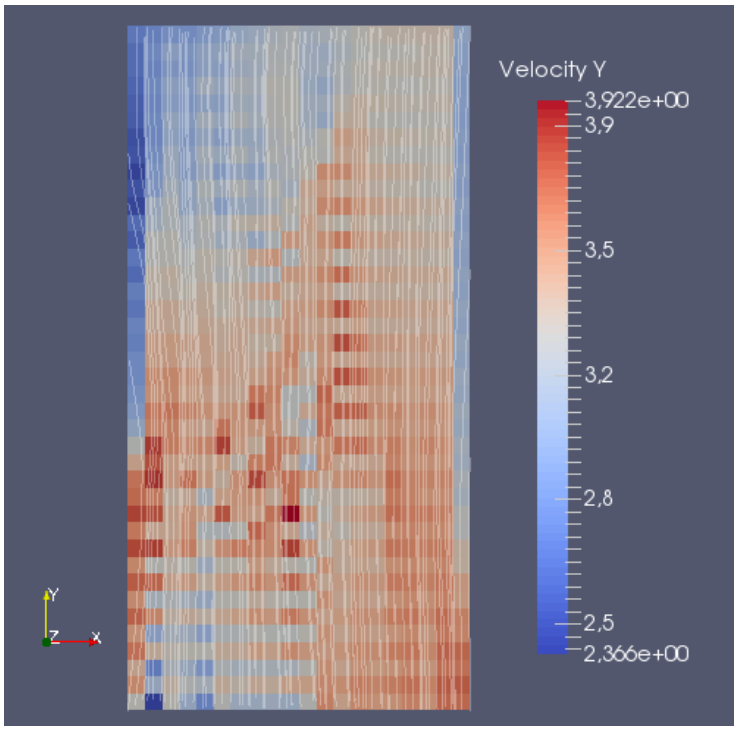

Figure 5.22: Two-phase - Velocity U_y, Low Mach Scheme

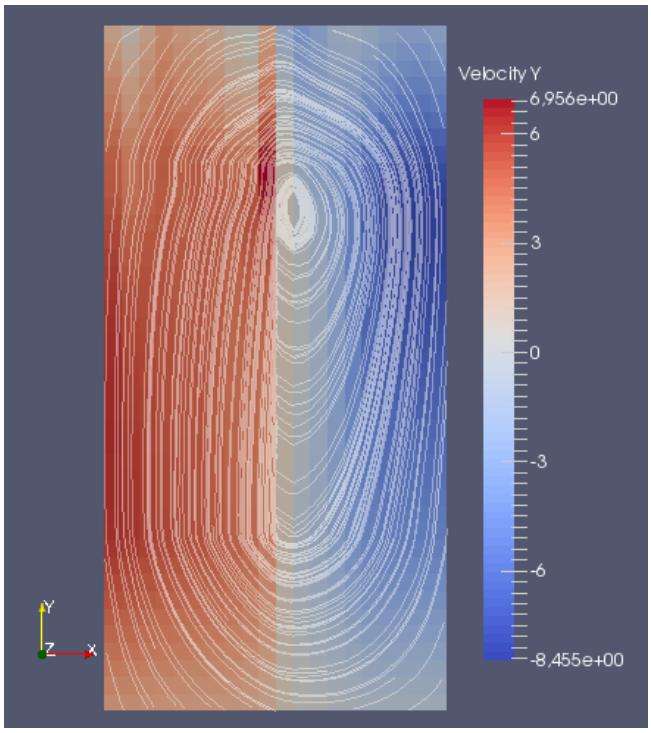

Figure 5.23: Two-phase - Velocity U_y, Upwind-well-balanced Scheme

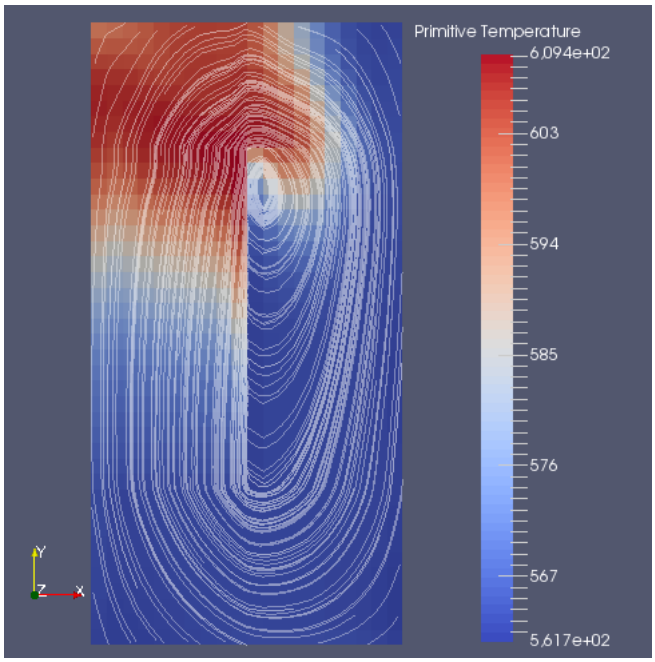

Figure 5.24: Two-phase - Temperature, velocity's streamlines U_y – Upwind-well-balanced Scheme

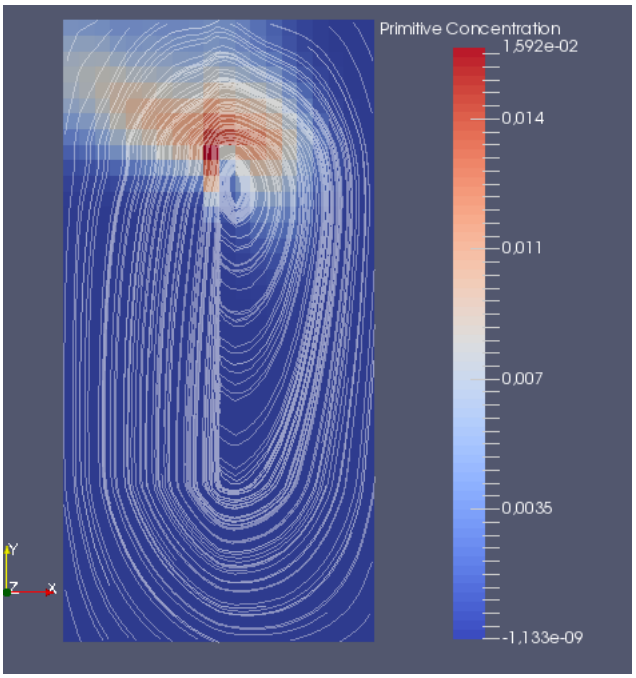

Figure 5.25: Two-phase - Concentration, velocity's streamlines U_y – Upwind-well-balanced Scheme

Two phase cases : inclined

We take the same configuration as in the section 5.3.1

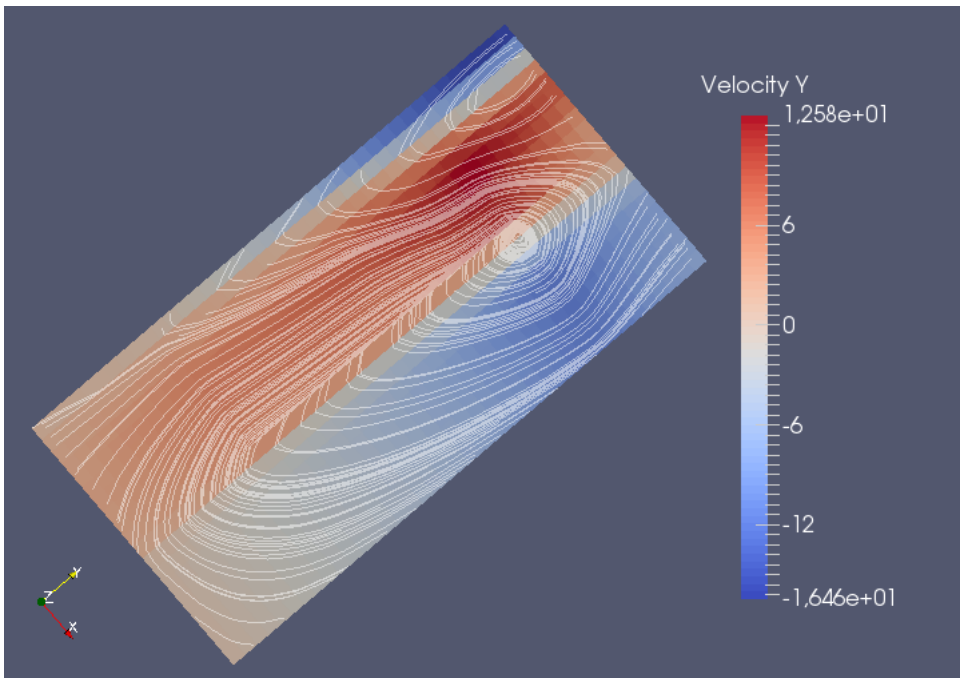

Figure 5.26: Two-phase - Inclined channel : Velocity U_y, Upwind-well-balanced Scheme

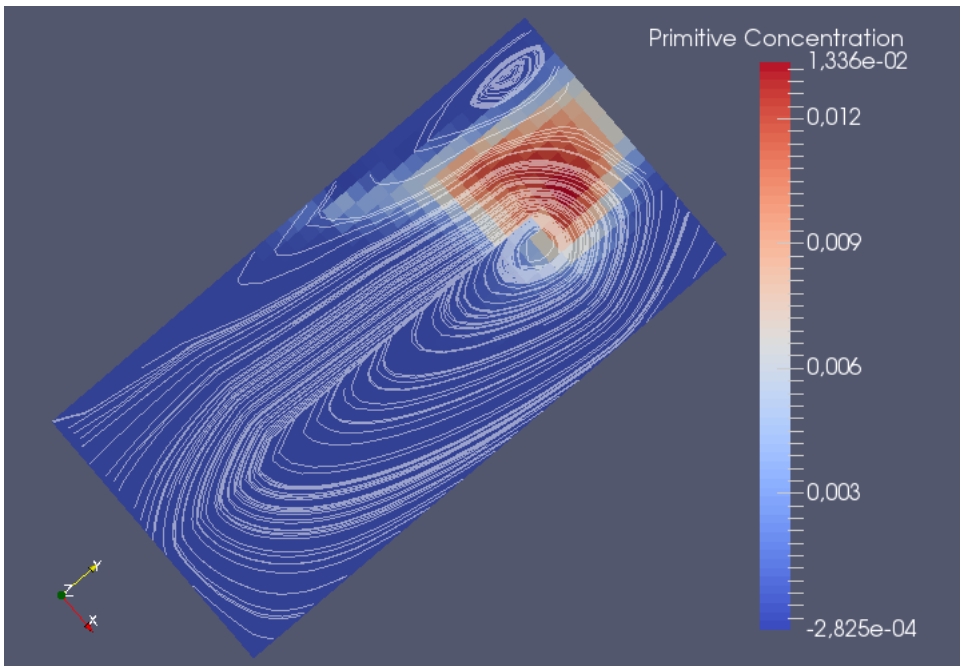

Figure 5.27: Two-phase - Inclined channel : Concentration, velocity's streamlines U_y – Upwind-well-balanced Scheme

The upwind well-balanced schemes were way more robust and the capture of the recirculation was not a problem unlike the low Mach number scheme which diverge around the wall.

These results show that the accuracy at low Mach number does not ensure a better behavior of the numerical methods

Chapter 6

Conclusion

Scientifically: :

The recirculation problem of the boiling flows in the parallel channels causes numerical and physical instabilities which make some calculations difficult to accomplish. This led us to test two new numerical approaches on this configuration.

The first scheme is the use of an upwinding in the source terms in order to obtain a well balanced scheme.
Whereas, the second is inspired by schemes on staggered mesh and is precise in low-Mach number.

After validating our development on an academic case test (Driven cavity), as we expected, we achieved better results with precise low-Mach number scheme(staggered).

- In a calculation on a pure thermal conduction (no source term), we have found that the staggered scheme is less diffusive than the upwind scheme.
- Whereas, for the calculations with a term of high thermal power, the results obtained by the upwind and well balanced scheme were much more robust and the recirculation capturing posed no difficulty, contrary to the low-Mach scheme which diverge around the bulkhead.

From all these results, we concluded that the precision in low March number doesn't guarantee a better behavior of numerical methods in the simulation of nuclear reactor's cores.

Futur work :

- It would be interesting to implement these numerical schemes in the industrial code ***FLICA 4***, to assess their robustness with more complex correlation and tabulated state laws.
- As a following up for this work would be an introduction of porosity and point losses to study the robustness of the proposed schemes. Analysis of the flows distribution problem between parallel channels would allow a validation of the physics of our numerical results.
- Analysis of the flows distribution problem between parallel channels would allow a validation of the physics of our numerical results.

On the personal level :

Having received a mathematical formation, this internship alowwed me to strengthen my knowledge in physics including mechanical fluids. It also allowed me to discover and use various libraries, open source, mesh handling, scientific computing and visualization. Furthermore, it was an opportunity to share and collaborate with lab engineers LGLS, which was very beneficial for me in particular.

I had also to overcome the obstacles due to safety standards, particularly, high at CEA Saclay.

Finally, I assume that the results of this training are highly positive because in addition to reaching the main objectives, I have got experienced and I have enlarged my professional environment by meeting scientists, researchers and engineers.

Chapter 7

Appendix

7.1 Time-discretization

7.1.1 Explicit scheme

We denote U_i^{n+1} as the conservative variables at time $n+1$, in the control cell C_i. The time explicit scheme is first-order in space and we can write the following discretization from the equation (3.2)

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{vol(C_i)} \sum_{j \in v(i)} S_{ij} \tilde{F}(U_i^n, U_j^n, \mathtt{n}_{ij}) + \Delta t S(U_i^n) \tag{7.1}$$

The equation (7.1) explicitly gives the conservative variables at time $n+1$, depending on these variables at time n. However, the use of this explicit formula needs small time steps which are limited by a CFL condition:

$$\triangle t \leq \frac{\triangle x}{|\lambda_M|}, \quad \triangle t \leq \frac{1}{K},$$

where λ_M is the highest eigenvalue of the matrixes $\nabla F \mathtt{n}_{ij}$.

If the velocities are equal, $\lambda_M = ||\mathtt{u}_m{}^{10}|| + a_m$ where a_m is the sound velocity of the two-phase mixture defined by:

$$a_m^2 = \kappa(H_m - \frac{1}{2}\mathtt{u}^2) + \chi + c_m \xi \tag{7.2}$$

with χ, ξ and κ defined in [6].

7.1.2 Implicit scheme

The use of implicit schemes enables to avoid stability conditions due to time steps and leads to more efficient numerical methods. The implicit scheme derived from (3.2) can be written under the following form:

$$U_i^n = U_i^{n-1} - \frac{\Delta t}{vol(C_i)} \sum_{j \in v(i)} S_{ij} \tilde{F}(U_i^n, U_j^n, \mathtt{n}_{ij}) + \Delta t S(U_i^n) \tag{7.3}$$

10. $\mathtt{u}_m = \frac{\alpha_l \rho_l \mathtt{u}_l + \alpha_v \rho_v \mathtt{u}_v}{\alpha_l \rho_l + \alpha_v \rho_v}$

The equation (7.3) is valid for all the control cells C_i, which define a non-linear system of equations that needs an iterative method and would be too heavy in terms of computing time.

Then, we have to solve a non-linear equation $f(\mathcal{U}) = 0$,
where $\mathcal{U} = (U_1, \ldots, U_N)$ and $f = (f_1, \ldots, f_N)$ where $f_i(\mathcal{U})$ is given by

$$f_i(\mathcal{U}) = \frac{U_i - U_i^n}{\triangle t} + \frac{1}{v_i} \sum_{j \in \nu(i)} s_{ij} \tilde{F}(U_i, U_j, \vec{n}_{ij}) - S(U_i).$$

The equation $f(\mathcal{U}) = 0$ is solved in a classic way using Newton's iterations

$$\nabla f(\mathcal{U}^k) \, (\mathcal{U}^{k+1} - \mathcal{U}^k) = -f(\mathcal{U}^k).$$

Each Newton's iteration needs a calculation of the gradient of f and the resolution of a linear system.
The exact calculation of the gradient of f is impossible because differentiating the function $D_{ij}(\mathcal{U})$ is very difficult. However, it is not necessary to calculate exactly the gradient to find the convergence of the Newton scheme. If the sequence $\mathcal{U}^k$ converges, then the limit $\mathcal{U}^\infty$ verifies $f(\mathcal{U}^\infty) = 0$.

Within Flica 4, in order to calculate an approximation of the gradient of the convection operator, we use the following relation

$$\sum_{j \in \nu(i)} s_{ij} \tilde{F}(U_i, U_j, \vec{n}_{ij}) = - \sum_{j \in \nu(i)} s_{ij} A_{ij}^-(U_i, U_j)(U_i - U_j), \tag{7.4}$$

Where $A_{ij}^- = \frac{1}{2}(A_{Roe}(U_i, U_j) - |A_{Roe}(U_i, U_j)|)$, where $A_{Roe}(U_i, U_j)$ is the Roe matrix of the flow F between the states U_i and U_j.

The equation (7.4) is written thanks to the definition of the Roe matrix(7.8), (3.9)

$$\tilde{F}(U_i, U_j, \vec{n}_{ij}) = F(U_i)\vec{n}_{ij} - A^-(U_i, U_j)\frac{U_i - U_j}{2},$$

Since the Roe matrix is unknown in the Flica4 model, we use the approximation

$$A_{ij}^- \approx A_{ij}^{m-}$$

with $A_{ij}^{m-} = \frac{1}{2}(A_{Roe}^m(U_i, U_j) - |A_{Roe}^m(U_i, U_j)|)$
where $A_{Roe}^m(U_i, U_j)$ is the Roe matrix of the flow F^m (équation 7.9) between the states U_i and U_j.

The final iterative scheme is written under the form:

$$\frac{U_i^{k+1}-U_i^k}{\triangle t}-\frac{1}{v_i}\sum_{j\in\nu(i)}s_{ij}A_{ij}^{m-}(U_i^k,U_j^k)((U_i^{k+1}-U_i^k)-(U_j^{k+1}-U_j^k))$$
$$-\nabla S(U_i^k)(U_i^{k+1}-U_i^k)=-\frac{U_i^k-U_i^n}{\triangle t}-\frac{1}{v_i}\sum_{j\in\nu(i)}s_{ij}\tilde{F}(U_i^k,U_j^k,\vec{n}_{ij})+S(U_i^k). \quad (7.5)$$

In the case of a face located at the border, we must know $U_b(U_i^k)$ and approach $U_b(U_i^{k+1})-U_b(U_i^k)$ from the equation (7.5).

The simplest process consists in dealing with the boundary conditions by using the linearization

$$U_b(U_i^{k+1})-U_b(U_i^k)\approx\nabla U_b(U_i^k)(U_i^{k+1}-U_i^k), \quad (7.6)$$

which needs the calculation of$U_b(U_i^k)$ and $\nabla U_b(U_i^k)$ for each boundary condition (appendix 7.2).

7.2 Boundary conditions

There are two types of boundary conditions in fluid mechanics: those which are linked with solid walls (normal zero flow) and those which are linked with the borders of the domain fixed by the user.
The process of boundary conditions in a finite volume scheme is tricky because the variables are defined in the volumes and not at the interfaces of the mesh.

To force the physical boundary conditions at the border of the domain, we add a fake mesh element (U_b, V_b) at the other side of the interface. This element is used only for computation: it allows the process of the limit interfaces as well as the interior interfaces (U,V). Once the state in the fake element is determined, all that remains is to calculate the numerical flow as if it was an interior interface.

The problem now consists in assigner to the fake mesh element a state such that the numerical flow calculated at the interface satisfies the required boundary conditions. From a mathematical point of view, the number of boundary conditions that we can impose must be the same number of inbound characteristics in the domain.

For the equations of FLICA 4, we consider four types of boundary conditions:

- Neumann :

$$V_b(V)=V,\quad U_b(U)=U$$

- Wall :

$$V_b : \begin{pmatrix} c_v \\ p \\ \mathtt{u} \\ T \end{pmatrix} \to \begin{pmatrix} c_v \\ p \\ -\mathtt{u} \\ T \end{pmatrix}, \quad U_b : \begin{pmatrix} \rho_m \\ \rho_m c_v \\ \rho_m \mathtt{u}_m \\ \rho_m E_m \end{pmatrix} \to \begin{pmatrix} \rho_m \\ \rho_m c_v \\ -\rho_m \mathtt{u}_m \\ \rho_m E_m \end{pmatrix}$$

- Input : concentration $c_{ve} = 1 - c_{le}$, velocity $\mathtt{u}_e$, temperature are all imposed T_e

$$V_b : \begin{pmatrix} c_v \\ p \\ \mathtt{u} \\ T \end{pmatrix} \to \begin{pmatrix} c_{ve} \\ p \\ \mathtt{u}_e \\ T_e \end{pmatrix}, \quad U_b : \begin{pmatrix} \rho \\ \rho c_v \\ \rho \mathtt{u} \\ \rho E \end{pmatrix} \to \begin{pmatrix} \rho_e \\ \rho_e c_{ve} \\ \rho_e \mathtt{u}_e \\ \rho_e E_e \end{pmatrix}$$

with

$$\begin{aligned} \rho_e &= \frac{\rho_v(p, T_e)\rho_l(p, T_e)}{\rho_v(p, T_e)c_{le} + \rho_l(p, T_e)c_{ve}} \\ E_e &= c_{ve}(e_v(T_e) + \frac{1}{2}|\mathtt{u}_{ve}|^2) + c_{le}(e_l(T_e) + \frac{1}{2}|\mathtt{u}_{le}|^2) \\ \mathtt{u}_{ve} &= \mathtt{u}_e + c_{le}\mathtt{u}_r(c_{ve}, \mathtt{u}_e, \rho_e) \\ \mathtt{u}_{ve} &= \mathtt{u}_e - c_{ve}\mathtt{u}_r(c_{ve}, \mathtt{u}_e, \rho_e). \end{aligned}$$

- Output: pressure p_s imposed

$$V_b : \begin{pmatrix} c_v \\ p \\ \mathtt{u} \\ T \end{pmatrix} \to \begin{pmatrix} c_v \\ p_s \\ \mathtt{u} \\ T \end{pmatrix}, \quad U_b : \begin{pmatrix} \rho \\ \rho c_v \\ \rho \mathtt{u} \\ \rho E \end{pmatrix} \to \begin{pmatrix} \rho_s \\ \rho_s c_v \\ \rho_s \mathtt{u} \\ \rho_s E \end{pmatrix}$$

with

$$\rho_s = \frac{\rho_v(p_s, T)\rho_l(p_s, T)}{\rho_v(p_s, T)c_l + \rho_l(p_s, T)c_v}.$$

7.3 Roe linearization

7.3.1 Definition of the Roe matrix

The Roe linearization is a technique introduced by P.Roe [21] to linearize accurately the systems of conservation laws under the form:

$$\partial_t U + \partial_x f(U) = 0, \tag{7.7}$$

We denote $A(U) = \nabla_U f(U)$ as the Jacobian matrix of the flow, which is diagonalizable in $\mathbb{R}$ in a complete base of real eigenvectors.
We can write (7.7) under a non-conservative form

$$\partial_t U + A(U)\partial_x U = 0$$

Note. We may notice that this form is correct for the regular solutions The Roe linearization is based on the following definition

Definition 7.3.1.1 *A matrix $A(U,V)$ which verfies*

1. *$A(U,V)$ is diagonalizable in $\mathbb{R}$ in a complete base of real eigenvectors,*
2. *For each (U,V) we have $f(U) - f(V) = A(U,V)(U-V)$,*
3. *For each U we have $A(U,U) = A(U)$,*

Is a Roe matrix associated to the system 7.7.

To calculate the flow between two neighbors i and j, the original idea of Roe in [21] was to linearize the problem (2.9) locally between the two states U_i and U_j. Roe is then looking for a linear interpolation of the flow F under the form

$$\tilde{F}(U) = F(U_i) + A_{Roe}(U_i, U_j)(U_i - U_j).$$

To obtain it, it is necessary to build a "Roe" matrix $A_{Roe}(U_i, U_j)$ which verifies

$$F(U_i) - F(U_j) = A_{Roe}(U_i, U_j)(U_i - U_j). \tag{7.8}$$

Although such a matrix always exists, it is not always easy to build it. That is the case for the model (2.9) where we need an approximated Roe matrix. For the calculation of the Roe matrix of the four-equation model, one must solve the equation (7.8) associated to the flow F. Since this calculation is difficult, the designers of Flica4 have overlooked the contribution of the gap velocity $\vec{u}_r$ in the flow F. This choice is adapted to the flows where the relative velocity $\vec{u}_r$ is negligible with respect to the average velocity s$\vec{u}_m$.

7.3.2 The Roe matrix in Flica4

The flow F (equation 2.10) can be divided into two contributions, the first one related to the average flow and the second one related to the relative movement:

$$F = F_m + F_r,$$

with

$$F_m(U) = {}^t \begin{pmatrix} \rho_m {}^t\vec{u}_m \\ \rho_m c_v {}^t\vec{u}_m \\ \rho_m \vec{u}_m \otimes \vec{u}_m + p\mathbb{I}_d \\ \rho_m H_m {}^t\vec{u}_m \end{pmatrix} \tag{7.9}$$

$$F_r(U) = {}^t \begin{pmatrix} 0 \\ \rho_m c_v c_l {}^t\vec{u}_r \\ \rho_m c_v c_l \vec{u}_r \otimes \vec{u}_r \\ \rho_m c_v c_l (H_v - H_l)^t \vec{u}_r \end{pmatrix}, \tag{7.10}$$

and

$$\begin{aligned} c_v &= \frac{\alpha_v \rho_v}{\alpha_v \rho_v + \alpha_l \rho_l}, \quad c_l = \frac{\alpha_l \rho_l}{\alpha_v \rho_v + \alpha_l \rho_l} \\ \rho_m &= \alpha_v \rho_v + \alpha_l \rho_l \\ \vec{u}_m &= \frac{\alpha_v \rho_v \vec{u}_v + \alpha_l \rho_l \vec{u}_l}{\alpha_v \rho_v + \alpha_l \rho_l} \\ H_m &= \frac{\alpha_v \rho_v H_v + \alpha_l \rho_l H_l}{\alpha_v \rho_v + \alpha_l \rho_l} = E_m + \frac{p}{\rho_m}. \end{aligned}$$

The software Flica4 overlooks the contribution associated to the relative component F_r of the convective flow F and uses the Roe matrix $A^m_{Roe}(U_i, U_j)$ associated to the average component F_m. By definition, this Roe matrix must:

- Be diagonalizable
- Verifies

$$F_m(U_i) - F_m(U_j) = A^m_{Roe}(U_i, U_j)(U_i - U_j)$$

- Verifies $A^m_{Roe}(U, U) = \nabla F_m(U)$, which is, in a one-dimensional space:

$$A^m_{Roe}(U, U) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -u_m c_v & u_m & c_v & 0 \\ \chi + (\frac{1}{2}\kappa - 1)u_m^2 & \xi & (2-\kappa)u_m & \kappa \\ (\chi + \frac{1}{2}\kappa u_m^2 - H)u_m & \xi u_m & H - \kappa u_m^2 & (\kappa+1)u_m \end{pmatrix}$$

With χ, ξ and κ defined by the relations mentioned in [6].

In the general case, when we have two states $U_i \neq U_j$, the Roe matrix $A^m_{Roe}(U_i, U_j)$ is

written under the form

$$A^m_{Roe}(U_i,U_j)=\begin{pmatrix} 0 & 0 & 1 & 0 \\ -\tilde{u}\tilde{c} & \tilde{u} & \tilde{c} & 0 \\ \tilde{\chi}+(\frac{1}{2}\tilde{\kappa}-1)\tilde{u}^2 & \tilde{\xi} & (2-\tilde{\kappa})\tilde{u} & \tilde{\kappa} \\ (\tilde{\chi}+\frac{1}{2}\tilde{\kappa}\tilde{u}^2-\tilde{H})\tilde{u} & \tilde{\xi}\tilde{u} & \tilde{H}-\tilde{\kappa}\tilde{u}^2 & (\tilde{\kappa}+1)\tilde{u} \end{pmatrix} \tag{7.11}$$

with

$$\begin{aligned} \tilde{u} &= \frac{\sqrt{\rho_{mi}}u_{mi}+\sqrt{\rho_{mj}}u_{mj}}{\sqrt{\rho_{mi}}+\sqrt{\rho_{mj}}} \\ \tilde{H} &= \frac{\sqrt{\rho_{mi}}H_{mi}+\sqrt{\rho_{mj}}H_{mj}}{\sqrt{\rho_{mi}}+\sqrt{\rho_{mj}}} \\ \tilde{c} &= \frac{\sqrt{\rho_{mi}}c_{vi}+\sqrt{\rho_{mj}}c_{vj}}{\sqrt{\rho_{mi}}+\sqrt{\rho_{mj}}}. \end{aligned}$$

The exact calculation of $\tilde{\chi},\tilde{\xi}$ and $\tilde{\kappa}$ is much more complex. We are going to define the following parameters

$$\begin{aligned} \tilde{\rho}_m &= \sqrt{\rho_{mi}\rho_{mj}} \\ \tilde{T} &= \frac{\sqrt{\rho_{mi}}T_i+\sqrt{\rho_{mj}}T_j}{\sqrt{\rho_{mi}}+\sqrt{\rho_{mj}}} \\ \tilde{p} &= \tilde{\rho}\tilde{H}-\frac{1}{2}\tilde{\rho}\tilde{u}^2-\tilde{\rho}(\tilde{c}e_v(\tilde{T})+(1-\tilde{c})e_l(\tilde{T})) \\ \tilde{\rho}_v &= \frac{\tilde{p}-p_{\infty v}}{(\gamma_v-1)e_v(\tilde{T})} \\ \tilde{\rho}_l &= \frac{\tilde{p}-p_{\infty l}}{(\gamma_v-1)e_l(\tilde{T})} \\ \tilde{\alpha}_v &= \frac{\tilde{c}\tilde{\rho}}{\tilde{\rho}_v}, \quad \tilde{\alpha}_l=1-\tilde{\alpha}_v \end{aligned}$$

And we will first use the following approximations derived from the equations ([6])

$$\begin{aligned} \tilde{\chi} &\approx \frac{\frac{\tilde{\alpha}_v\tilde{\rho}_v c_v^0+\tilde{\alpha}_l\tilde{\rho}_l c_l^0}{\tilde{\rho}_l}-e_l(\tilde{T})(\frac{\tilde{\alpha}_v c_v^0}{e_v(\tilde{T})}+\frac{\tilde{\alpha}_l c_l^0}{e_l(\tilde{T})})}{\frac{\tilde{\alpha}_v}{\tilde{p}-p_{\infty v}}+\frac{\tilde{\alpha}_l}{\tilde{p}-p_{\infty l}}} \\ \tilde{\xi} &\approx \frac{(\tilde{\alpha}_v\tilde{\rho}_v c_v^0+\tilde{\alpha}_l\tilde{\rho}_l c_l^0)\frac{\tilde{\rho}_l-\tilde{\rho}_v}{\tilde{\rho}_v\tilde{\rho}_l}-(e_v(\tilde{T})-e_l(\tilde{T}))(\frac{\tilde{\alpha}_v c_v^0}{e_v})}{\frac{\tilde{\alpha}_v}{\tilde{p}-p_{\infty v}}+\frac{\tilde{\alpha}_l}{\tilde{p}-p_{\infty l}}} \\ \tilde{\kappa} &\approx \frac{\frac{\tilde{\alpha}_v c_v^0}{e_v(\tilde{T})}+\frac{\tilde{\alpha}_l c_l^0}{e_l(\tilde{T})}}{\frac{\tilde{\alpha}_v}{\tilde{p}-p_{\infty v}}+\frac{\tilde{\alpha}_l}{\tilde{p}-p_{\infty l}}}. \end{aligned}$$

7.3.3 Spectrum of the Roe matrix

In a one-dimensional space, the Roe matrix $A^m_{Roe}(U_i, U_j)$ (equation 7.11) has four eigenvalues

$$\lambda_1 = \tilde{u} - \tilde{a}, \quad \lambda_2 = \tilde{u}, \quad \lambda_3 = \tilde{u}, \quad \lambda_4 = \tilde{u} + \tilde{a},$$

where $\tilde{a}$ is the sound velocity of the two-phase mixture, given by

$$\tilde{a}^2 = \tilde{\kappa}(\tilde{H} - \frac{1}{2}\tilde{u}^2) + \tilde{\chi} + \tilde{c}\tilde{\xi}.$$

The four eigenvectors associated on the left and on the right are

$$\vec{\tilde{r}}_1 = \begin{pmatrix} 1 \\ \tilde{c} \\ \tilde{u} - \tilde{a} \\ \tilde{H} - \tilde{u}\tilde{a} \end{pmatrix} \quad , \quad \vec{\tilde{l}}_1 = \frac{1}{2\tilde{a}^2} \begin{pmatrix} \tilde{\chi} + \frac{1}{2}\tilde{\kappa}\tilde{u}^2 + \tilde{a}\tilde{u} \\ \tilde{\xi} \\ -(\tilde{\kappa}\tilde{u} + \tilde{a}) \\ \tilde{\kappa} \end{pmatrix},$$

$$\vec{\tilde{r}}_4 = \begin{pmatrix} 1 \\ \tilde{c} \\ \tilde{u} + \tilde{a} \\ \tilde{H} + \tilde{u}\tilde{a} \end{pmatrix} \quad , \quad \vec{\tilde{l}}_4 = \frac{1}{2\tilde{a}^2} \begin{pmatrix} \tilde{\chi} + \frac{1}{2}\tilde{\kappa}\tilde{u}^2 - \tilde{a}\tilde{u} \\ \tilde{\xi} \\ -(\tilde{\kappa}\tilde{u} - \tilde{a}) \\ \tilde{\kappa} \end{pmatrix},$$

$$\vec{\tilde{r}}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ -\frac{\tilde{\xi}}{\tilde{\kappa}} \end{pmatrix} \quad , \quad \vec{\tilde{l}}_2 = \begin{pmatrix} -\tilde{c} \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

$$\vec{\tilde{r}}_3 = \begin{pmatrix} 1 \\ \tilde{c} \\ \tilde{u} \\ \frac{1}{2}\tilde{u}^2 - \tilde{c}\tilde{\xi} - \tilde{\chi} \end{pmatrix} \quad , \quad \vec{\tilde{l}}_3 = -\frac{1}{\tilde{a}^2} \begin{pmatrix} \tilde{\chi} + \frac{1}{2}\tilde{\kappa}\tilde{u}^2 - \tilde{a}^2 \\ \tilde{\xi} \\ -\tilde{\kappa}\tilde{u} \\ \tilde{\kappa} \end{pmatrix}.$$

7.3.4 Calculation of the "upwinding"

For the Roe scheme ([21]), the upwind matrix D_{ij} (equation 3.9) should be equal to $|A_{Roe}(U_i, U_j)|$, the absolute value of the Roe matrix associated to the normal flow $F_{\vec{n}_{ij}} = F^m_{\vec{n}_{ij}} + F^r_{\vec{n}_{ij}}$. However, since this one is tough to calculate, the designers of Flica4 have rather chosen

$$D_{ij} = |A^m_{Roe}(U_i, U_j)| = \sum_k |\lambda_k| \vec{\tilde{l}}_k \otimes \vec{\tilde{r}}_k, \tag{7.12}$$

where $A^m_{Roe}(U_i, U_j)$ is the Roe matrix of mixture (equation 7.11) associated to the flow of mixture $F^m_{\vec{n}_{ij}}$ whose spectrum $(\lambda_k, \vec{\tilde{l}}_k, \vec{\tilde{r}}_k)$ is given at section 7.3.3.

The choice of this approximation of the Roe matrix is adapted to flows which have quite the same velocity where the relative velocity $\vec{u}_r$ is negligible with respect to the average velocity$\vec{u}_m$. The calculation of the "upwinding" is done within Flica4, using the equation 7.12 and therefore the spectral decomposition of $A^m_{Roe}(U_i, U_j)$ (appendix 7.3.3) . It is then limited in case the two phases have equal velocities. In our model, we use the technique of polynomial interpolation which does not need the calculation of the eigenvectors of$A^m_{Roe}(U_i, U_j)$, but only its eigenvalues. We hope that this choice will allow us to use the exact Roe matrix $D_{ij} = |A_{Roe}(U_i, U_j)|$ later for the calculation of the "upwinding" and then to deal with the non negligible velocity gaps better, especially those from countercurrent flows.

7.3.5 Pressure correction

We are going to adopt a multidimensional approach because the pressure corrections can bring a significant improvement of the accuracy of the scheme. Once the approached Roe matrix is calculated, the original Roe flow through a face with a normal $\vec{n}_{ij}$ can be written

$$\tilde{F}(U_i, U_j, \vec{n}_{ij}) = \frac{F(U_i) + F(U_j)}{2}\vec{n}_{ij} - \quad |A^m_{Roe}(U_i, U_j, \vec{n}_{ij})|\frac{U_i - U_j}{2}, \tag{7.13}$$

and denoting $\tilde{u}_n = \vec{\tilde{u}} \cdot \vec{n}_{ij}$ the normal velocity to the face ij

$$|A^m_{Roe}(U_i, U_j, \vec{n}_{ij})| \quad = \quad \left| \begin{pmatrix} 0 & 0 & {}^t\vec{n}_{ij} & 0 \\ -\tilde{u}_n\tilde{c} & \tilde{u}_n & \tilde{c}^t\vec{n}_{ij} & 0 \\ (\tilde{\chi} + \frac{1}{2}\tilde{\kappa}|\vec{\tilde{u}}|^2)\vec{n}_{ij} - \tilde{u}_n\vec{\tilde{u}} & \tilde{\xi}\vec{n}_{ij} & \vec{\tilde{u}} \otimes \vec{n}_{ij} + \tilde{u}_n\mathbb{I}_d - \tilde{\kappa}\vec{n}_{ij} \otimes \vec{\tilde{u}} & \tilde{\kappa}\vec{n}_{ij} \\ (\tilde{\chi} + \frac{1}{2}\tilde{\kappa}\tilde{u}^2 - \tilde{H})\tilde{u}_n & \tilde{\xi}\tilde{u}_n & \tilde{H}{}^t\vec{n}_{ij} - \tilde{\kappa}\tilde{u}_n{}^t\vec{\tilde{u}} & (\tilde{\kappa}+1)\tilde{u}_n \end{pmatrix} \right|$$

However, the Roe scheme, even if it is stable, may create accuracy problem when it comes to low Mach number. Indeed, the pressure at the interface for the flow (7.13) is equal to

$$\tilde{p} = \frac{1 + \frac{\tilde{u}}{\tilde{a}}}{2}p_i + \frac{1 - \frac{\tilde{u}}{\tilde{a}}}{2}p_j - \frac{\tilde{a}}{2}(\rho_{mi}\vec{u}_{mi} - \rho_{mj}\vec{u}_{mj}) \cdot \vec{n}_{ij} + O(||\rho_{mi}\vec{u}_{mi} - \rho_{mj}\vec{u}_{mj}||^2).$$

When the Mach number $\frac{\tilde{u}}{\tilde{a}}$ is nearly zero, the contribution of the "upwinding" $\frac{\tilde{a}}{2}(\rho_{mi}\vec{u}_{mi} - \rho_{mj}\vec{u}_{mj}) \cdot \vec{n}_{ij}$ stays relevant because $|\vec{\tilde{u}}| \ll \tilde{a}$ and the numerical diffusion is too high. The designers of Flica4 have then introduced “pressure corrections” to increase the accuracy.

The updated flow may be written

$$\tilde{F}(U_i, U_j, \vec{n}_{ij}) = \frac{F(U_i) + F(U_j)}{2}\vec{n}_{ij} - D_{ij}\frac{U_i - U_j}{2} + \frac{\tilde{a}(\rho_{mi}\vec{u}_{mi} - \rho_{mj}\vec{u}_{mj}) \cdot \vec{n}_{ij}}{2} \begin{pmatrix} 0 \\ 0 \\ \vec{n}_{ij} \\ 0 \end{pmatrix},$$

which is the same thing than adding a new contribution D^p_{ij} to the "upwinding" D_{ij} with

$$D^p_{ij} = \frac{\tilde{a}}{2} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \vec{n}_{ij} \otimes \vec{n}_{ij} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

If we want the scheme to stay stable with this pressure correction, we should just accept a CFL below 0.5 instead of 1.

7.4 Handling of the steady states

To show the importance of the upwinding of the stiff source terms, we will consider the following system of 1D stead balance laws

$$\partial_x F(U) = S(U, x), \tag{7.14}$$

where S is a vector whose all the components are positive, which is especially the case of the drift model of CoreFlows. The solution U is then such that all the components of $F(U)$ are increasing. This generally leads to the monotony of U and the absence of vibrations in the solution. This property of monotony of the components of U is not always preserved at the discrete level and parasitic vibrations may appear, especially when S has discontinuity points. Let's consider the following classic upwind scheme:

$$F_{i+\frac{1}{2}} = \frac{F_i + F_{i+1}}{2} + \texttt{signe}(\bar{A})\frac{F_i - F_{i+1}}{2}, \tag{7.15}$$

where $\bar{A}$ is an approximation of the jacobian diagonalizable matrix A, whose calculation is not always exact. In this case,$\bar{A}$ is the Roe matrix associated to the normal flow $F_{\mathtt{n}}$ appearing in the equation (3.1) such that

$$F_{\mathtt{n}} = F^m_{\mathtt{n}} + F^r_{\mathtt{n}}$$

where
$F^m_{\mathtt{n}}$ is the flow of mixture [(appendix 7.3]

$F^r_{\mathtt{n}}$ is the flow related to the relative movement

However, since $\bar{A}$ is hard to calculate, the designers of Flica4 have chosen:

$$\bar{A} = A^m_{Roe}$$

This choice is adapted to flows where the relative velocity $\mathtt{u}_r$ is negligible with respect to the average velocity $\mathtt{u}_m$

7.4.1 Characterization of the steady state

The steady state associated to the system 3.1 is defined by the following equation :

$$\partial_x F(U^{stat}(x,t)) = S(U,x). \tag{7.16}$$

For the steady problem (7.16) to have a unique solution, we suppose that the matrix A is invertible and especially $\mathtt{signe}(\bar{A}) * \mathtt{signe}(\bar{A}) = \mathbb{I}_d$.

If the source term S is positive, *i.e* all it components are positive, the equation (7.16) involves that all the components of $F(U)^{stat}$ are increasing, and then should not oscillate. If a component of S is equal to zero, then the associate component of $F(U)$ should be constant.

We want to approach the solutions of (7.16) numerically and still keep the monotony of the flows at the discrete level. For that purpose, the discretized steady state must satisfy the following equation

$$\frac{F_{i+1} - F_i}{\triangle x} = S_{i+\frac{1}{2}}, \tag{7.17}$$

where $S_{i+\frac{1}{2}}$ corresponds to $S(U,x,t)$ and ensures the conservation of the steady states if $S_{i+\frac{1}{2}} \geq 0$

The steady state of (3.12) is characterized by:

$$\frac{F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}}{\triangle x} = S_i, \tag{7.18}$$

From (7.15), (7.17) and (7.18) we can deduct the relation between S_i and $S_{i+\frac{1}{2}}$.

By replacing (7.15) in (7.18), we obtain :

$$\frac{\mathbb{I}_d - \mathtt{signe}(\bar{A})}{2}\frac{F_{i+1} - F_i}{\triangle x} + \frac{\mathbb{I}_d + \mathtt{signe}(\bar{A})}{2}\frac{F_i - F_{i-1}}{\triangle x} = S_i.$$

And from (7.17)

$$\frac{\mathbb{I}_d - \mathtt{signe}(\bar{A})}{2} S_{i+\frac{1}{2}} + \frac{\mathbb{I}_d + \mathtt{signe}(\bar{A})}{2} S_{i-\frac{1}{2}} = S_i. \tag{7.19}$$

Yet, we have $\texttt{signe}(\bar{A}) * \texttt{signe}(\bar{A}) = \mathbb{I}_d$ and by multiplying (7.19) by $\frac{\mathbb{I}_d - \texttt{signe}(\bar{A})}{2}$, we obtain:

$$\frac{\mathbb{I}_d - \texttt{signe}(\bar{A})}{2} S_{i+\frac{1}{2}} = \frac{\mathbb{I}_d - \texttt{signe}(\bar{A})}{2} S_i. \tag{7.20}$$

By multiplying (7.19) by $\frac{\mathbb{I}_d + \texttt{signe}(\bar{A})}{2}$ and by replacing i by $i+1$, we obtain

$$\frac{\mathbb{I}_d + \texttt{signe}(\bar{A})}{2} S_{i+\frac{1}{2}} = \frac{\mathbb{I}_d + \texttt{signe}(\bar{A})}{2} S_{i+1}. \tag{7.21}$$

From (7.20) and (7.21), we can deduct:

$$S_{i+\frac{1}{2}} = \frac{I_d + \texttt{signe}(\bar{A})}{2} S_{i+1} + \frac{I_d - \texttt{signe}(\bar{A})}{2} S_i. \tag{7.22}$$

7.4.2 Upwind scheme with a centered source term

The classic upwind scheme with the centered source term is written

$$\frac{\mathrm{d}U_i}{\mathrm{d}t}(t) + \frac{F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}}{\triangle x} = S(U_i).$$

The steady state verifies

$$\frac{F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}}{\triangle x} = S(U_i).$$

Applying the same method than previously 7.4.1, we obtain

$$\frac{F_{i+1} - F_i}{\triangle x} = \frac{S(U_i) + S(U_{i+1})}{2} + \texttt{signe}(\bar{A}) \frac{S(U_{i+1}) - S(U_i)}{2}. \tag{7.23}$$

(7.23) is a consistent discretization of the equation at the steady state but does not keep the properties of monotony of the solution when one or several components of S are negative or have a fixed sign. If S is almost constant, we are not far from a monotone solution. However, for stiff source terms, the scheme can strongly breach the monotony because we cannot control the sign of $\frac{S(U_i)+S(U_{i+1})}{2} + \texttt{signe}(\bar{A}) \frac{S(U_{i+1})-S(U_i)}{2}$.

7.4.3 Fully upwind scheme with an upwind source term

The discrete system (7.24) comes naturally from the analysis of the Riemann problem with a source term for a hyperbolic system

$$\frac{\mathrm{d}U_i}{\mathrm{d}t}(t) + \frac{F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}}{\triangle x} = \frac{1}{2}\left(S_{i+\frac{1}{2}} + S_{i-\frac{1}{2}}\right), \tag{7.24}$$

where

$$\begin{array}{rcl} S_{i-\frac{1}{2}} & = & \frac{S(U_{i-1})+S(U_i)}{2} + \mathtt{signe}(\bar{A})\frac{S(U_{i-1})-S(U_i)}{2} \\ S_{i+\frac{1}{2}} & = & \frac{S(U_i)+S(U_{i+1})}{2} + \mathtt{signe}(\bar{A})\frac{S(U_i)-S(U_{i+1})}{2} \\ S(U_i) & = & S(U_i, x_i, t). \end{array}$$

In this case, the source term and the scheme are both upwind.
We obtain a discretization of the source term which is not trivial. That implies the upwinding at the interface of the neighbors. The first term of the equality (7.24) may be different from $S(U_i)$ if the source term is stiff. The discrete steady solution of (7.24) solves

$$\frac{F^{stat}_{i+\frac{1}{2}} - F^{stat}_{i-\frac{1}{2}}}{\triangle x} = \frac{1}{2}\left(S_{i+\frac{1}{2}} + S_{i-\frac{1}{2}}\right). \tag{7.25}$$

Applying the same method than in the section 7.4.1, we rewrite (7.25) by using (7.15) as follows :

$$\frac{\mathbb{I}_d - \mathtt{signe}(\bar{A})}{2}\frac{F_{i+1} - F_i}{\triangle x} + \frac{\mathbb{I}_d + \mathtt{signe}(\bar{A})}{2}\frac{F_i - F_{i-1}}{\triangle x}$$

$$=$$

$$\frac{\mathbb{I}_d - \mathtt{signe}(\bar{A})}{2}\frac{S(U_i) + S(U_{i+1})}{2} + \frac{\mathbb{I}_d + \mathtt{signe}(\bar{A})}{2}\frac{S(U_{i-1}) + S(U_i)}{2}.$$

By multiplying (7.26) by $\frac{\mathbb{I}_d - \mathtt{signe}(\bar{A})}{2}$, we obtain

$$\frac{\mathbb{I}_d - \mathtt{signe}(\bar{A})}{2}\frac{F_{i+1} - F_i}{\triangle x} = \frac{\mathbb{I}_d - \mathtt{signe}(\bar{A})}{2}\frac{S(U_i) + S(U_{i+1})}{2}. \tag{7.26}$$

By multiplying (7.26) by $\frac{\mathbb{I}_d + \mathtt{signe}(\bar{A})}{2}$, and replacing i by $i+1$, we obtain

$$\frac{\mathbb{I}_d + \mathtt{signe}(\bar{A})}{2}\frac{F_{i+1} - F_i}{\triangle x} = \frac{\mathbb{I}_d + \mathtt{signe}(\bar{A})}{2}\frac{S(U_i) + S(U_{i+1})}{2}. \tag{7.27}$$

From (7.26) and (7.27) we can deduct

$$\frac{F_{i+1} - F_i}{\triangle x} = \frac{S(U_i) + S(U_{i+1})}{2}, \tag{7.28}$$

7.28 is a uniform discretization of the steady equation (7.16) which verifies the same properties of monotony. We can then conclude that the upwinding of the source term involves the non-oscillation of the steady state.

Bibliography

[1] M. Ishii. Thermo-Fluid Dynamic Theory of Two-Phase Flow., volume 22 of Direction des études et recherches d'électricité de France. Eyrolles, Paris, 1975.

[2] J. M. Masella, I. Faille and T. Gallouët, "On an Approximate Godunov Scheme", Intl. J. ComputationalFluid Dynamics, 1999, Vol. 12, pp. 133-149.

[3] D.A. Drew and S.L. Passman. Theory of Multicomponent Fluids., volume 135 of Applied Mathematical Sciences. Springer, New York, 1998

[4] J.M Delhaye, M. Giot, and M.L. Riethmuller. Thermohydraulics of Two-Phase Systems for Industrial Design and Nuclear Engineering, volume 5 of A Von Karman Iinstitute Book. Mc Graw Hill Book Compagny, Hemisphere Publishing Corporation, 1981.

[5] K. Halaoua. Quelques Solveurs pour les opérateurs de convection et leur application à la mécanique des fluides diphasiques. PhD thesis, ENS Cachan, 1998

[6] T. Sid Abdelkader.Maquettage et test du code de thermohydraulique diphasique FLICA4, Rapport de PFE, SUP' Galilée -Université Paris 13, 2015

[7] Angelo Murrone. Modèles bi-fluides à six et sept équations pour les écoulements diphasiques a 'faible nombre de Mach. Earth Sciences. Université de Provence - Aix-Marseille I, 2003. French

[8] Projet "Ce sont Des Mathématiques Appliquées à la THermohydraulique": `http://cdmath.jimdo.com/`

[9] Toolbox du projet CDMATH : `https://github.com/PROJECT-CDMATH/CDMATH`

[10] Petsc 3.4.5 - Parallel storage and manipulation of large sparse matrices : `http://www.mcs.anl.gov/petsc/`

[11] A. S. Shieh, V. H. Ransom, R. Krishnamurthy, Validation of numerical techniques in RELAP5/MOD3, RELAP5/MOD3 code manual volume 6, October 1994

[12] `http://www.casl.gov/COBRA-TF.shtml`

[13] `http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/Fluid+Dynamics+Products/ANSYS+CFX`

[14] J. J. jeong, H. Y. Yoon, I. K. Park, H. K. Cho, J. Kim, A semi-implicit numerical scheme for transient two-phase flows, Nuclear Engineering and Design 236 (2008) 3403-3412

[15] I. Toumi, A. Bergeron, D. Gallo, E. Royer, D. Caruge, "FLICA-4: a three-dimensional two-phase flow computer code with advanced numerical methods for nuclear applications," Nuclear Engineering and Design, Volume 200, Issues 1-2, August 2000, Pages 139-155.

[16] Ph. Fillion, I.Toumi. Flica-4 V.1 Calcul des flux diffusifs dans le cas d'une géométrie non-structurée, Rapport CEA, DMT 95/335

[17] S. Dellacherie, The Bichteler-Dellacherie Theorem, `https://almostsure.wordpress.com/2011/03/28/the-bichteler-dellacherie-theorem/`

[18] D. Bestion, "The Physical Closure Laws in The CATHARE Code, Nuclear Engineering and Design," vol. 124, pp. 229-245, 1990.

[19] N. Méchitoua, M. Boucker, J. Laviéville, J.-M. Hérard, S. Pigny, and G. Serre. An Unstructured Finite Volume Solver for Two-Phase Water/Vapour Flows Modelling Based on an Elliptic Oriented Fractional Step Method. In NURETH 10, International Meeting on Nuclear Reactor Thermal-Hydraulics, Seoul, South Korea, 2003.

[20] Propriétés thermophysiques des systèmes fluides, `http://webbook.nist.gov/chemistry/fluid/`

[21] Ph. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys., 43, 357, 1981

[22] "Upwind methods for hyperbolic conservation laws with source terms", A. Bermudez, E. Vazquez, Comp. Fluids, vol 23, issue 8, pp. 1049-1071,1994

[23] M. Ndjinga, T.-P.-K. Nguyen, C. Chalons, Numerical simulation of an incompressible two-fluid model, Springer Proc. Math. & Stat., Vol. 78, Finite Volumes for Complex Applications FVCA7, 2014

[24] T.-H. Dao, M. Ndjinga, F. Magoules, Comparaison of Upwind and Centered Schemes for Low Mach Number Flows, Finite Volumes for Complex Applications VI - Problems & Perspectives, Springer Proceedings in Mathematics 4, 2011

[25] R. J. LeVeque, Finite Volume Methods for Hyperbolic Problems, Cambridge University Press, 2002

[26] The open-source integration platform for numerical simulation `www.salome-platform.org`

[27] The open-source, parallel data analysis and visualization application www.paraview.org

[28] The Visualization ToolKit www.vtk.org

[29] The open-source integration platform for numerical simulation, : www.eclipse.org

[30] The open-source integration platform for numerical simulation, : www.codeblocks.org

[31] Parallel storage and manipulation of large sparse matrices www.mcs.anl.gov/petsc

[32] Simplified Wrapper and Interface Generator : http://www.swig.org

[33] HDF5 - Data model and file format of large volume : http://www.hdfgroup.org/HDF5

[34] DOXYGEN - Generate documentation from source code : http://www.doxygen.org

[35] CMAKE - Open-source build test and package software : http://www.cmake.org

[36] https://fr.wikibooks.org/wiki/Programmation_Qt/Qt_Designer

List of Figures

Contents