INTERNSHIP REPORT

Development of numerical schemes for 1D Euler's equations resolution oriented towards shock capture

---

M1 Internship
May 15th - September 1rst, 2023
Cadarache Center, Saint-Paul-lez-Durance

Student
RENARD Maxime

Supervisors
CHAUVIN Rémi
FRANCESCATTO Jérôme
PONS Kévin

Acknowledgements

I would like to cheerfully thank Rémi Chauvin, Jérôme Francescatto and Kévin Pons for their presence, their enthusiasm when speaking about my advancement and obviously their patience when I had some trouble debugging my code (especially Rémi). I have gained quite a lot of knowledge thanks to them, both on mathematical or physics aspects, as well as on some useful tricks to make sure code is working or even about results presentation and knowledge transmission.

It seems also important to me to thank Laurent Saas, head of the LMAG, who enables the laboratory to work in a good atmosphere and to facilitate scientific talks, as well as the rest of the laboratory team which has been deeply welcoming and interesting to spend time with.

They are not part of the LMAG team but work at the CEA too : Antoine Llor and Antoine Gerschenfeld. The first one, specialist of staggered schemes, helped me a lot with his remarks and critics on my work ; he took time on his holidays to do so. The second one, member of the `TRUST` team, gave to my tutors and myself a lot of ways and possibilities to explore in our scheme implementation, which have guided my research during this internship.

Finally, a big thumbs up to all the other students that were there, who have built a really nice team energy and with which the midday break was quite entertaining.

I am grateful for these opportunities and I am convinced that these 3.5 months have been very exciting and useful for my formation, personal interests and future professional life.

Contents

Introduction

This report summarizes the work I have done during my internship on the nuclear research center of the french Atomic Energy Commission (CEA), Cadarache. It spanned on three and a half month during which I was supervised by Rémi Chauvin, Jérôme Francescatto and Kévin Pons.

With the approaching arrival of sodium-cooled fast reactors (SFR, fr. RNR-Na), the need of numerical tools for risks assessing and for getting a better understanding of underlying concepts arises, such as for vapor explosions and corium/sodium interaction. An aim of the french Atomic Energy Commission is to keep a hand on the knowledge and the expertise that such topics require. For this reason, a choice has been made to use a pre-existing tool developed by the CEA, `TRUST` , and to upgrade it towards these abilities.

`TRUST` comes from a CFD platform, `TrioU` which has been split in 2015 into `TrioCFD` and `TRUST` when `TrioU` was transferred from Grenoble to Saclay CEA center. In some sense, `TrioCFD` is a whole CFD software for incompressible fluids whereas `TRUST` is the numerical core of the platform. Both of these codes implemented at first incompressible 3D models, that slowly tend to be adapted towards compressible ones as the needs arise through some phenomena in nuclear reactors.

`TRUST` implements so called Mark-and-Cells schemes (MAC schemes) where velocities are defined at mesh interfaces; it also makes use of some implicit parts within the schemes. It is a heavy code as it is massively parallel, able to deal with 2D, 3D, multi-fluid and multi-physics problems. Moreover it uses some custom `C++` tools as it has first been developed without using STL. For instance, pointers, references, vectors and other objects are redefined. Finally, no mathematics, neither numerical nor informatics documentation on its multi-fluid aspects exist as it has been implemented until the end of 2021.

Because of all these reasons, `TRUST` is not suited for performing numerical investigation directly inside of it. This is why I worked on a 1D simple `C++` code that Rémi Chauvin had started to develop for my arrival.

The aims of this internship are thus to perform some numerical investigation, both on MAC schemes and on `TRUST` numerical methods, to try to reproduce or even improve them. The main mission being to adapt it for compressible problems (ultimately heavy shocks) while minimizing the changes to bring to the schemes as it might one day be implemented inside `TRUST` platform.

Chapter 1

Commissariat à l'énergie Atomique

In this chapter, we are going to introduce what the *Commissariat à l'énergie Atomique* (CEA, eng. *Atomic Energy Commission*), both about its history, how, when and why it has been created as well as its inner branching and its domains of activities.

1.1 History

Established in 1945, shortly after World War II, the CEA was created with the aim of coordinating and promoting scientific research and peaceful applications of atomic energy.

Figure 1.1: Original order [1] of creation of the CEA on September 18th, 1945

At the beginning of its existence, the CEA focused a lot on fundamental research and the development of nuclear reactors for electricity generation and defense. Including the renowned Saclay Nuclear Research Centre, many research facilities were established to support these activities. The CEA quickly gained international recognition for its research and development in the field of nuclear energy.

During the 1950s and 1960s, the CEA was also involved in military research projects, particularly in the field of nuclear weapons [2]. These activities were part of France's nuclear deterrence program. While military research was an integral part of the CEA's activities at the time, the emphasis was always on the peaceful uses of atomic energy.

In the next decades, it expanded its expertise areas to diversify in many research fields, such as defense or medical applications. It engaged studies on nuclear fusion, radioactive waste treatment, nuclear safety, defense, and many other related areas. The CEA also played a prominent role in

international cooperation, collaborating with other research institutions and international organizations to share knowledge and technologies in the field of atomic energy.

By contributing to the building and operation of nuclear power plants, the commission has been a real key player in the promotion of nuclear energy in France. It has also worked on innovative technologies such as fast neutron reactors and nuclear fusion reactors to meet future energy needs.

Nowadays, the commission is still a major player in the field of atomic energy and scientific research in France as it strives to develop safer, cleaner, and more efficient technologies for the future of nuclear energy, as well as many other technologies mentioned further.

1.2 Activities

The Atomic Energy Commission (CEA) is involved in a wide range of activities and technologies related to atomic energy. These activities aim to harness the power of nuclear reactions for various beneficial purposes while ensuring safety, security, and sustainability. All this information has been found on the firm's website (url for CEA research domains). Here are some of the key areas in which the CEA operates.

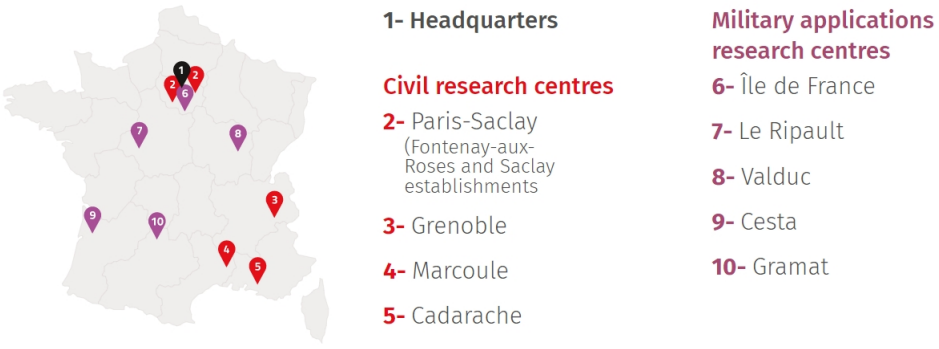


Figure 1.2: CEA Centers locations (presentation webpage)

Nuclear defense The CEA is involved in various aspects of nuclear defense, including the design, manufacture, and maintenance of nuclear warheads for French airborne and seaborne nuclear forces. They also design and build nuclear reactors for French navy's ships, submarines, and aircraft carriers, providing ongoing support and maintenance for these reactors. Additionally, the CEA contributes to national and international security by assisting authorities in combating nuclear proliferation, terrorism and disarmament. They also provide expertise in conventional defense activities, focusing on weapon effects and vulnerability. Its simulation program, supercomputers, radiographic facilities and laser systems help ensuring safety, reliability, and certification of nuclear weapons and propulsion systems, with avoiding the need of doing real scale and real life tests as it has been done in the past and which is forbidden nowadays.

Energy The CEA's takes part in the development of low-carbon power systems that integrate nuclear and renewable energies. They explore various aspects such as innovative methods for low-carbon power generation, efficient energy storage systems, effective control and conversion techniques and resource management within a circular economy framework. Their expertise in these areas benefits both the French government and industry stakeholders. The CEA's research spans a wide range of topics including current and future nuclear reactors, the nuclear fuel cycle, solar power, hydrogen

technologies, small modular reactors (SMRs), and thermonuclear fusion as it takes part in the ITER project ; some component have been tested on the CEA's tokamak : WEST. Additionally, they investigate energy system control and management, including future energy networks, stationary energy storage systems, and sustainable mobility solutions. The CEA also actively contributes to circular economy by optimizing the nuclear fuel cycle, exploring technologies for a carbon circular economy, and promoting responsible resource management.

Nuclear Medicine and biology Nuclear medicine involves the application of radioactive isotopes for diagnostic imaging, therapy, and research purposes. The CEA conducts research in radiobiology and toxicology to understand the effects of radiation, radionuclides, and nanoparticles on humans and the environment. Their studies contribute to radiation protection standards and medical radiotherapy advancements. Pathogenesis research explores cellular processes and disease mechanisms, aiding in the development of therapeutic and diagnostic strategies. The CEA also focuses on innovative diagnostics, therapies and nanomedicine, aiming to improve patient care through biomarker discovery, targeted treatments and regenerative medicine. Overall, the CEA's research enhances scientific knowledge, medical practices and solutions for health challenges. The main part of these activities are located in Grenoble.

Matter and universe The CEA conducts theoretical and simulation-based research in diverse areas of physics. In nanoscience, it explores the properties and applications of materials at the atomic scale. Material research covers structural analysis, photonics, superconductivity and magnetism. Nuclear physics studies atomic nuclei, including exotic and superheavy ones, with implications for astrophysics. The CEA contributes to instrument design and testing of theoretical models for particle physics. Finally, it is involved in large international astrophysical observation programs across different energy ranges. The CEA excels in instrumentation, designing experiments and utilizing high-performance computing.

Electricity production The production of electricity through nuclear power plants is a key area of emphasis for the CEA. The commission is responsible for overseeing the licensing, regulation, and enforcement of safety standards for these facilities, ensuring their efficient, reliable and secure operation. One could think about "France Relance" program initiated by french government in September 2020. Furthermore, the CEA actively supports research and development efforts in advanced reactor designs, with the goal of improving the safety and efficiency of nuclear power generation.

Nuclear Safety and Security Ensuring the safety and security of nuclear facilities and materials is a critical aspect of the CEA's responsibilities. It establishes and enforces stringent safety regulations and guidelines to minimize the risks associated with nuclear activities. The commission also collaborates with law enforcement agencies and international organizations to prevent nuclear weapons proliferation and to safeguard against illicit nuclear activities.

Nuclear Science Education and Outreach The CEA plays a crucial role in educating the public about atomic energy and dispelling misconceptions surrounding nuclear technology. It provides educational programs, public forums and informational resources to increase public awareness and understanding of nuclear science and its benefits. The CEA also plays major role in training young scientists and engineers, which is a non negligible mission, thanks to which I have had the pleasure having my internship at this firm.

Overall, the Atomic Energy Commission undertakes a wide range of activities and utilizes various technologies to harness the potential of atomic energy for the betterment of society. While ensuring safety and security, the commission strives to promote innovation, sustainable practices and the peaceful use of nuclear technology.

1.3 The severe accident modeling laboratory (LMAG)

The LMAG (fr. *Laboratoire de Modélisation de Accidents Graves*) specializes in modeling severe accidents, for instance those in which the reactor's core has melted, resulting in the formation of corium with high temperature. This is a significant concern because it can potentially breach the tank containing the uranium fuel, trigger vapor explosions upon contact with the surrounding water, and eventually reach the underlying layer of concrete that supports the facility. Such incidents can occur, for instance, in pressurized water reactors, which are the predominant type of reactors found in France's current nuclear power plants.

Reactor designs aim to prevent reaching such critical stages, but there is always a possibility of natural disasters or design-related issues. A prime example is the *Chernobyl* catastrophe in the late 1980s and the meltdowns of reactors 1, 2, and 3 at *Fukushima Daiichi* following a tsunami in 2011. This underscores the importance of gaining a comprehensive understanding of these issues to effectively address them and make informed decisions regarding the security measures that should be implemented in case of their occurrence.

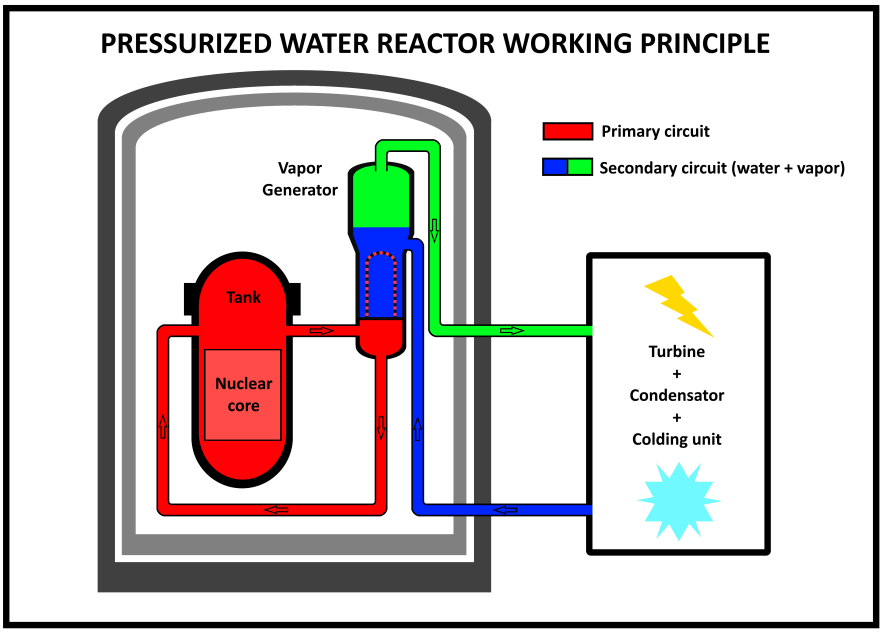


Figure 1.3: Simplified view of how a pressurized reactor manages to produce electricity

The LMAG conducts studies on these types of events using a numerical approach. Replicating accident conditions is challenging and costly. As a result, programs are being developed to simulate these configurations. The laboratory utilizes various codes to examine each stage of severe accidents across different reactor types, ranging from core meltdown to corium's interaction with concrete. For example, the `SIMMER` code models the core melting's candeling effect and how corium interacts with the tank in RNR-Na reactors. `MC3D` is employed to simulate the interaction between corium and water, including vapor explosions. The `SCONE` code, deriving from `TRUST` and which forms the basis of this internship, focuses on CFD which is mainly associated with vapor explosions and core cooling.

These codes play a crucial role in modeling configurations that may give rise to issues and they are also essential for assessing the safety of existing reactors. Utilizing these codes requires extensive expertise in numerical methods for implementing the governing equations, as well as a deep understanding of neutronics, physics and mathematics.

Chapter 2

Euler's 1D equations

A general overview and some comprehension keys are going to be given to the reader about what these equations are, what they model, where they come from and why they are meaningful in this internship context. Note that the homogeneous system is going to be introduced, but a source term for gravity could be added to the momentum and total energy equations.

2.1 Mono-fluid equations

2.1.1 Governing equations

For a chosen material, we denote its density ρ, velocity v, specific internal energy ε and finally its pressure p. Euler's equations[3] are then formulated as :

$$\frac{\partial(\rho)}{\partial t} + \frac{\partial(\rho v)}{\partial x} = 0 \qquad \text{(Mass/Density)}$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho v^2)}{\partial x} + \frac{\partial(p)}{\partial x} = 0 \qquad \text{(Momentum)}$$

$$\frac{\partial(\rho \varepsilon)}{\partial t} + \frac{\partial(\rho \varepsilon v)}{\partial x} + p\frac{\partial(v)}{\partial x} = 0 \qquad \text{(Internal Energy)}$$

One can find on the left the names we choose to give to these equations in this report ; it corresponds to the quantity they act one.

Note that these equations are set as functional relations which could refer to distributions. This will be the case if dealing with shocks, the density could be a Heaviside function for example. Note that as the velocity is not forced to have null divergence, this system describes a compressible fluid.

The system has 4 variables : ρ, ε, v, p. In fact, pressure will be deduced from density and internal energy thanks to an equation of state :

$$p = \mathcal{P}(\rho, \varepsilon) \tag{2.1}$$

which is determined empirically with many constants, defined when coping with thermodynamics. Section 2.3 will provide details about the closure we chose : the Stiffened Gas equation of state.

Remark At first, these equations were designed to represent perfect fluid movements[4] with only the Momentum equation. Euler's work has then been extended in 1822 by Henri Navier who introduced viscosity term in the momentum equation[5] and few other mathematicians and physicists have contributed to it, such as George Gabriel Stokes. These equations are well suited for adiabatic flows, meaning no heat flux exists between the domain of study and the outside environment.

2.1.2 Conservativity property

Euler's equations are build around density, momentum and total energy conservation, meaning they preserve their quantity over the domain. This ensures a natural rule which is that matter or movement cannot emerge of vanish from nothingness. Conservativity is a property enabling numerical schemes to converge towards the right weak solution.

Definition / Property

1. A smooth quantity f is said to be conserved if its integral over a closed domain Ω changes in time at the speed at which it is transported through the boundary of the domain in which it is contained i.e.

$$\frac{\partial}{\partial t}\int_{\Omega}\phi(t,x)dx = -\int_{\partial\Omega}F(\phi(t,x))\cdot n dx \tag{2.2}$$

2. An equation (E) is said to be conservative for a function ϕ if there exists a smooth functional F such that equation (E) is equivalent to

$$\frac{\partial(\phi)}{\partial t} + \nabla_x \cdot F(\phi) = 0 \tag{2.3}$$

with space derivative adapted to the number of dimensions.

A proof of equivalency of these two definitions is made in appendix A assuming some measurability and smoothness properties on the flux and ϕ. It is then clear that Euler's equations are conservative for density and momentum. On the other hand, due to its shape, the internal energy equation is not conservative (variable outside the gradient).

For instance, if the domain is closed with null flux on the boundary, the total mass and momentum should be preserved. This concept thus ensures that when no source or sink exists, the system is isolated. This fundamental principle guarantees the stability and integrity of the system, regardless of the specific dynamics of the underlying equations.

2.1.3 Total energy formulation

As seen before, Euler's equation are not conservative for internal energy. However, they do preserve total energy. This can be proven building the kinetic energy equation and combining it to the internal energy one. This is a method to build total specific energy as the sum of internal and kinetic energies. The procedure is detailed in appendix B and leads to the total energy equation expressed as :

$$\frac{\partial(\rho e)}{\partial t} + \frac{\partial(\rho e v)}{\partial x} + \frac{\partial(pv)}{\partial x} = 0 \tag{2.4}$$

where it is considered that total specific energy is $e = \varepsilon + \frac{1}{2}v^2$. Due to its shape, this equation is a conservation equation with some transport of the total energy inside the domain.

Within Euler's equations system, it is possible to replace the specific internal energy equation with this total specific energy one. This would mean that the total energy becomes an unknown, the internal energy will be derived from it, meaning the nature of the problem changes. We would then say we work under a *total energy formulation.*

2.2 Multi-fluid Euler's equations

In this chapter, we work under a multi-fluid formulation, meaning we could have mixtures of different fluids occurring. We add an exponent k to designate a particular specie.

2.2.1 Governing equations

Still working under a eulerian framework, the multi-fluid version of the equations is formulated as follows.

$$\frac{\partial(\alpha^k\rho^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k v^k)}{\partial x} = 0 \tag{2.5}$$

$$\frac{\partial(\alpha^k\rho^k v^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k v^k v^k)}{\partial x} + \alpha^k\frac{\partial(p^k)}{\partial x} = 0 \tag{2.6}$$

$$\frac{\partial(\alpha^k\rho^k\varepsilon^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k\varepsilon^k v^k)}{\partial x} + p^k\left(\frac{\partial(\alpha^k)}{\partial t} + \frac{\partial(\alpha^k v^k)}{\partial x}\right) = 0 \tag{2.7}$$

where $\rho^k, v^k, p^k, \varepsilon^k$ still denote the density, velocity, pressure and internal specific energy of the fluid, but specifically design specie k ones. Variable α^k denotes the volumic fraction of the fluid, and should always sum to 1 when adding over its index k as it is a barycenter. It is the proportion of specie k, as a function of space and time.

One major difference with mono-fluid equations is that the momentum is no longer preserved. This comes from the possible conversion of momentum from a specie to another.

2.2.2 Closure of the system

In their current formulation, assuming there are K different species, the equations have $5K$ unknowns meaning the system has be closed. Two paradigms exists which both introduce an equation of state

$$p^k = \mathcal{P}(\rho^k, \varepsilon^k) \tag{2.8}$$

Baer-Nunziato approach[6] In this approach, all p^k could be different so as they are deduced from internal energy and density, K closure relations are to be found. These relations in fact concern the volumic fraction by adding an advection equation on it

$$\frac{\partial(\alpha^k)}{\partial t} + v^k\frac{\partial(\alpha^k)}{\partial x} = S^k(p^1, \dots, p^K) \tag{2.9}$$

where S^k is a well chosen source term including differences of pressure between species. For instance, if $K = 2$ it is expressed as $S^1 = p^2 - p^1$ and $S^2 = p^1 - p^2$. For numerical sake, a weighting coefficient might be added before S^k to perform the so-called *pressure relaxation* to avoid the system blowing up. Finally, to make sure the volumic fraction is barycentric, α^K is deduced from other volumic fractions. If S^k is chosen not to be null, a fractional step method should be used[7] and some additional source terms should be added in momentum and energy equation[8, 9].

Isobaric approach The main difference with the Baer-Nunziato approach is that it is assumed $p^1 = \cdots = p^K$. This means there are actually $4K + 1$ unknowns. Using Euler's equations with the addition of the equation of state provides $4K$ closure relations, meaning one misses. The barycentric condition on the volumic fraction is then added to close the system.

2.2.3 Total energy formulation

As for the mono-fluid case, a multi-fluid version of the total specific energy can be derived. It uses the very same tricks as for the mono-fluid case ; detailed calculations can be found in appendix C. The obtained formulation is

$$\frac{\partial(\alpha^k\rho^k e^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k e^k v^k)}{\partial x} + \frac{\partial(\alpha^k p^k v^k)}{\partial x} + p^k\frac{\partial(\alpha^k)}{\partial t} = 0 \tag{2.10}$$

On the contrary to mono-fluid total specific energy equation, this one is note conservative. In fact there could be some energy exchanges between two different fluids both in kinetic and internal energy. However, when summing over index k meaning we consider species as a whole corps, it is clear that we get back the mono-fluid energy equation, which is conservative. This means that over the domain, the global total energy is preserved, but not at the scale of each specie.

2.3 Stiffened gas equation

It has been mentioned multiple times that an equation of state will be used to close the system. The so-called *Stiffened-Gas* equation[10, 11] is chosen as it is widely used in multi-component flow problem applications, easy to implement and well suited for shocks[12]. The formula is given as follows :

$$p^k = (\gamma^k - 1)\varepsilon^k\rho^k - \gamma^k p^k_* \tag{2.11}$$

where p^k, ε^k, ρ^k are the pressure, specific internal energy and density of the fluid, p^k_* is empirically determined and represents attraction between the fluid's molecules and γ^k the adiabatic index of the fluid. It is a linearized version of Mie–Grüneisen equation of state[13] and its *stiffened* character comes from p^k_* which enables to capture tension as it was first designed for solids.

2.4 Artificial viscosity

When dealing with shocks, there is no weak solution uniqueness as solutions are defined almost everywhere. This means a choice shall be made, generally on the only physically consistent solution, the entropic one, when viscosity tends towards zero. However, adding a hand-crafted term to mimic viscosity in order to stabilize numerical schemes is possible. Such a term is expressed as the linear combination of a linear and a quadratic member [14]

$$-\alpha\rho c\Delta x\frac{\partial(v)}{\partial x} + \beta(\gamma+1)\frac{\rho}{4}\Delta x^2\left(\min(0, \frac{\partial(v)}{\partial x})\right)^2 \tag{2.12}$$

where α, β are some weighting coefficients for the linear combination, c is the sound speed in the fluid, ρ the fluid's density, v^k its velocity and γ its corresponding adiabatic coefficient.

As it is explained by Debord[15], the quadratic term, weighted by β performs a diffusion of the shock in order to slightly smooth it and avoid numerical artefacts, it stabilizes strong shocks. Its min term has to role to activate it only when a node is in a compression state. On the other hand, the linear term, weighted by α has the role to dump non physical oscillatory behaviours provoked by the quadratic term.

Ideally, as `TRUST` has no need of such a tool, we wish our schemes will not need it, even though it is now an available tool for shock stabilization.

Chapter 3

Discretization of the equations

In this chapter, the finite volume method will be used. In one dimension, it is similar to the use of finite differences apart from the fact that the space step is multiplying the time derivative instead of dividing discrete spatial gradients.

3.1 Framework and notations

We are going to discretize the equations with the so-called Marker-and-Cell (MAC) scheme presented by F. H. Harlow and J. E. Welch [16]. It consists in defining density, volumic fraction, pressure and specific internal energy at the cells' centers, and to define the velocity at interfaces. This has the advantage to make the pressure gradient be of second order[1]. Consider that all the unknowns are function of time and space.

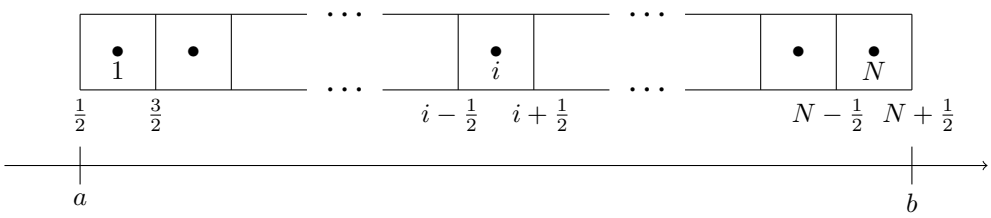
3.1.1 Spatial mesh

On the spatial domain, one define a mesh with cells indexed from 1 to N, which center position is denoted x_i for cell i. One defines the interfaces (dual mesh) thanks to their position denoted $x_{i+\frac{1}{2}}$ or $x_{i-\frac{1}{2}}$. Thanks to these notations, one can define their length :

- for cell i (c_i): $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$
- for interface $i+\frac{1}{2}$ ($c_{i+\frac{1}{2}}$): $\Delta x_{i+\frac{1}{2}} = x_{i+1} - x_i$

To denote the integration over a cell's spatial domain (from its left interface to its right one), we will denote the cell as c_i. In a same way, to denote integration over an interface's spatial domain, we will denote the interface as $c_{i+\frac{1}{2}}$ or $c_{i-\frac{1}{2}}$.

Figure 3.1: Cells and interfaces illustration



With this discretization, you can see that $x_{\frac{1}{2}} = a$ and that $x_{N+\frac{1}{2}} = b$ meaning there is indeed one more interface than cell centers.

[1] Write a Taylor development under its integral form : $p_{i+1} = p_i + \Delta x_{i+\frac{1}{2}} \int_{c_{i+\frac{1}{2}}} \frac{\partial(p)}{\partial x} dx$.

In order to make the scheme more readable, we will choose the following notations :

$$\alpha_i^k = \frac{1}{\Delta x_i}\int_{c_i} \alpha^k dx \qquad \rho_i^k = \frac{1}{\Delta x_i \alpha_i^k}\int_{c_i} \alpha^k\rho^k dx$$
$$\varepsilon_i^k = \frac{1}{\Delta x_i \alpha_i^k \rho_i^k}\int_{c_i} \alpha^k\rho^k\varepsilon^k dx \qquad p_i^k = \frac{1}{\Delta x_i \alpha_i^k}\int_{c_i} \alpha^k p^k dx$$

These ones seems to be a good choice as they enable to write the finite volume discretization almost at sight for Euler's multi-fluid equations, while keeping the equations readable and making sens mathematically. Finally, note that these objects are actually functions of time.

3.1.2 Interpolation method

The use of finite volumes make some undefined terms appear such as velocity on cells centers or density at interfaces. The choice of *upwind* interpolation is made as Euler equations represent advection. This means that the interpolation follows the natural direction of information propagation. For instance, it is proved in appendix D that the choice of centered interpolation for $1D$ advection equation is unconditionally unstable, meaning this choice guarantees stability of the scheme, in the Von-Neumann analysis sense.

In order to be explicit, for the quantity $\Phi \in \{\alpha, \rho, \varepsilon, p\}$ which is defined at the cells center, interface value is interpolated with an upwind method :

$$\Phi_{i+\frac{1}{2}}^k = \begin{cases} \Phi_i & \text{if } v_{i+\frac{1}{2}}^k \geq 0 \\ \Phi_{i+1} & \text{if } v_{i+\frac{1}{2}}^k < 0 \end{cases} \tag{3.1}$$

Finally, for the velocity, which is defined at the interface, it will be interpolated at cells centers with an upwind manner :

$$v_i^k = \begin{cases} v_{i-\frac{1}{2}}^k & \text{if } v_{i+\frac{1}{2}}^k \geq 0 \\ v_{i+\frac{1}{2}}^k & \text{if } v_{i+\frac{1}{2}}^k < 0 \end{cases} \tag{3.2}$$

Note that depending on the nature of some terms, it might be clever to choose other kinds of interpolation. This will be investigated in the result chapter.

3.2 Density and internal energy equations discretization

Here, we derive the scheme for the density equation and the internal energy. First, recall the multi-fluid mass and energy equations :

$$\begin{cases} \frac{\partial(\alpha^k\rho^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k v^k)}{\partial x} & = 0 \\ \frac{\partial(\alpha^k\rho^k\varepsilon^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k\varepsilon^k v^k)}{\partial x} + p^k\left(\frac{\partial(\alpha^k)}{\partial t} + \frac{\partial(\alpha^k v^k)}{\partial x}\right) & = 0 \end{cases} \tag{3.3}$$

3.2.1 Time discretization

We choose to work with explicit Euler forward method for simplicity sake. Time step can change over time but will respect the relation $t^{n+1} = t^n + \Delta t$ where t^n is the n-th iteration of the scheme and Δt the quantity of time to go to next time.

For density, time derivative is expressed as follows :

$$\int_{c_i} \frac{\partial(\alpha^k\rho^k)}{\partial t} dx = \Delta x_i \frac{\alpha_i^{k,n+1}\rho_i^{k,n+1} - \alpha_i^{k,n}\rho_i^{k,n}}{\Delta t} \tag{3.4}$$

The energy time derivative is discretized as :

$$\int_{c_i} \frac{\partial(\alpha^k \rho^k \varepsilon^k)}{\partial t} dx = \Delta x_i \frac{\alpha_i^{k,n+1} \rho_i^{k,n+1} \varepsilon_i^{k,n+1} - \alpha_i^{k,n} \rho_i^{k,n} \varepsilon_i^{k,n}}{\Delta t} \tag{3.5}$$

and volumic fraction time derivative is written as :

$$\int_{c_i} p^k \frac{\partial(\alpha^k)}{\partial t} dx = \Delta x_i p_i^k \frac{\alpha_i^{k,n+1} - \alpha_i^{k,n}}{\Delta t} \tag{3.6}$$

where the assumption is that the pressure is constant over each cell, which is why one can extract the pressure for the integral.

3.2.2 Space discretization

We still use finite volumes and integrate over an arbitrary cell c_i as this is the place where pressure, density, volumic fraction and internal energy are defined.

The space derivative coming from the density equation is treated as :

$$\int_{c_i} \frac{\partial(\alpha^k \rho^k v^k)}{\partial x} dx = \left[\alpha_j^k \rho_j^k v_j^k\right]_{j=i-\frac{1}{2}}^{j=i+\frac{1}{2}} = \alpha_{i+\frac{1}{2}}^k \rho_{i+\frac{1}{2}}^k v_{i+\frac{1}{2}}^k - \alpha_{i-\frac{1}{2}}^k \rho_{i-\frac{1}{2}}^k v_{i-\frac{1}{2}}^k \tag{3.7}$$

and finally the two space terms from the internal energy equation are expressed as :

$$\int_{c_i} \frac{\partial(\alpha^k \rho^k \varepsilon^k v^k)}{\partial x} dx = \left[\alpha_j^k \rho_j^k \varepsilon_j^k v_j^k\right]_{j=i-\frac{1}{2}}^{j=i+\frac{1}{2}} = \alpha_{i+\frac{1}{2}}^k \rho_{i+\frac{1}{2}}^k \varepsilon_{i+\frac{1}{2}}^k v_{i+\frac{1}{2}}^k - \alpha_{i-\frac{1}{2}}^k \rho_{i-\frac{1}{2}}^k \varepsilon_{i-\frac{1}{2}}^k v_{i-\frac{1}{2}}^k \tag{3.8}$$

$$\int_{c_i} p^k \frac{\partial(\alpha^k v^k)}{\partial x} dx = p_i^k \left[\alpha_j^k v_j^k\right]_{j=i-\frac{1}{2}}^{j=i+\frac{1}{2}} = p_i^k \left(\alpha_{i+\frac{1}{2}}^k v_{i+\frac{1}{2}}^k - \alpha_{i-\frac{1}{2}}^k v_{i-\frac{1}{2}}^k\right) \tag{3.9}$$

3.2.3 Summary

To sum up, the final scheme for the density equation is :

$$\Delta x_i \frac{\alpha_i^{k,n+1} \rho_i^{k,n+1} - \alpha_i^{k,n} \rho_i^{k,n}}{\Delta t} + \left[\alpha_{i+\frac{1}{2}}^k \rho_{i+\frac{1}{2}}^k\right]_u v_{i+\frac{1}{2}}^k - \left[\alpha_{i-\frac{1}{2}}^k \rho_{i-\frac{1}{2}}^k\right]_u v_{i-\frac{1}{2}}^k = 0 \tag{3.10}$$

The final scheme for the specific internal energy equation is :

$$\begin{aligned} &\Delta x_i \frac{\alpha_i^{k,n+1} \rho_i^{k,n+1} \varepsilon_i^{k,n+1} - \alpha_i^{k,n} \rho_i^{k,n} \varepsilon_i^{k,n}}{\Delta t} \\ &+ \left[\alpha_{i+\frac{1}{2}}^k \rho_{i+\frac{1}{2}}^k \varepsilon_{i+\frac{1}{2}}^k\right]_u v_{i+\frac{1}{2}}^k - \left[\alpha_{i-\frac{1}{2}}^k \rho_{i-\frac{1}{2}}^k \varepsilon_{i-\frac{1}{2}}^k\right]_u v_{i-\frac{1}{2}}^k \\ &+ p_i^k \left(\Delta x_i \frac{\alpha_i^{k,n+1} - \alpha_i^{k,n}}{\Delta t} + \left[\alpha_{i+\frac{1}{2}}^k\right]_u v_{i+\frac{1}{2}}^k - \left[\alpha_{i-\frac{1}{2}}^k\right]_u v_{i-\frac{1}{2}}^k\right) = 0 \end{aligned} \tag{3.11}$$

Colored terms are those which shall be interpolated with an upwind method. Note that they are **interpolated together as conserved quantities**.

3.3 Non conservative momentum equation discretization

Multiple discretization of the momentum equation will be shown. Only non conservative ways of proceedings are used here as the multi-fluid expression is non conservative. Recall the momentum equation is :

$$\frac{\partial(\alpha^k \rho^k v^k)}{\partial t} + \frac{\partial(\alpha^k \rho^k v^k v^k)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{3.12}$$

3.3.1 Reformulation of the equation

In the will of imitating TRUST discretization which is called *semi-conservative* formulation, rather than seeing the equation as dealing on the momentum, we shape it to be on the velocity. For this purpose, on can reformulate the equation by expanding derivatives containing velocity :

$$\alpha^k \rho^k \frac{\partial(v^k)}{\partial t} + v^k \frac{\partial(\alpha^k \rho^k)}{\partial t} + \alpha^k \rho^k v^k \frac{\partial(v^k)}{\partial x} + v^k \frac{\partial(\alpha^k \rho^k v^k)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{3.13}$$

Note that orange terms express the density equation with a product by velocity, meaning they can be deleted. The final form of the equation we stop at is :

$$\alpha^k \rho^k \frac{\partial(v^k)}{\partial t} + \alpha^k \rho^k v^k \frac{\partial(v^k)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{3.14}$$

After a discussion with A. Gernschenfeld from the TRUST team, we chose to express $v^k \frac{\partial(v^k)}{\partial x}$ in three different ways. We give them a surname which has nothing to do with intrinsic mathematical properties but with how these terms look :

- Non-conservative : $v^k \frac{\partial(v^k)}{\partial x}$
- Semi-conservative : $\frac{\partial((v^k)^2)}{\partial x} - v^k \frac{\partial(v^k)}{\partial x}$, formulation used in TRUST
- Quasi-conservative : $\frac{1}{2} \frac{\partial((v^k)^2)}{\partial x}$

To conclude on the reformulation, there are now 3 different ways to treat the momentum equation depending the spatial derivative :

$$\alpha^k \rho^k \frac{\partial(v^k)}{\partial t} + \alpha^k \rho^k v^k \frac{\partial(v^k)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{NCV}$$

$$\alpha^k \rho^k \frac{\partial(v^k)}{\partial t} + \alpha^k \rho^k \frac{1}{2} \frac{\partial((v^k)^2)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{QCV}$$

$$\alpha^k \rho^k \frac{\partial(v^k)}{\partial t} + \alpha^k \rho^k \left(\frac{\partial((v^k)^2)}{\partial x} - v^k \frac{\partial(v^k)}{\partial x} \right) + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{SCV}$$

As mentioned in the literature[17, 18] the interest of having multiply possibilities for the momentum equation comes from the behaviour of the numerical error. In fact, even if continuous expressions are equal, they might not have same numerical precision and properties when running heterogeneous multi-fluid versions of the problem.

3.3.2 Time discretization

We still use forward Euler scheme to process time integration :

$$\int_{c_{i+\frac{1}{2}}} \alpha^k \rho^k \frac{\partial(v^k)}{\partial t} dx = \Delta x_{i+\frac{1}{2}} \alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}} \frac{v^{k,n+1}_{i+\frac{1}{2}} - v^{k,n}_{i+\frac{1}{2}}}{\Delta t} \tag{3.15}$$

Beware, here the method straddles two half cells as the velocity is defined only at interfaces. In other words, it is applied on the **dual mesh**.

3.3.3 Space discretization

First we discretize the pressure term before showing what is done on velocity related terms :

$$\int_{c_{i+\frac{1}{2}}} \alpha^k \frac{\partial(p^k)}{\partial x} dx = \alpha^k_{i+\frac{1}{2}} \left(p^k_{i+1} - p^k_i \right) \tag{3.16}$$

We see that defining the interpolation method was useful as here the volumic fraction is needed at an interface. Let us now derive the scheme for the different possible expression of the equation :

- For orange term in equation NCV, applying finite volumes on a dual cell $i+\frac{1}{2}$ provides :

$$\int_{c_{i+\frac{1}{2}}} \alpha^k \rho^k v^k \frac{\partial(v^k)}{\partial x} dx = \alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}} v^k_{i+\frac{1}{2}} \left(v^k_{i+1} - v^k_i\right) \tag{3.17}$$

- For the orange term from QCV, which looks like a conservative term, we get :

$$\int_{c_{i+\frac{1}{2}}} \alpha^k \rho^k \frac{1}{2} \frac{\partial((v^k)^2)}{\partial x} dx = \frac{\alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}}}{2} \left((v^k_{i+1})^2 - (v^k_i)^2\right) \tag{3.18}$$

- For the orange term in equation SCV, finite volumes provide the relation :

$$\int_{c_{i+\frac{1}{2}}} \alpha^k \rho^k \left(\frac{\partial((v^k)^2)}{\partial x} - v^k \frac{\partial(v^k)}{\partial x}\right) dx = \alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}} \left[(v^k_{i+1})^2 - (v^k_i)^2 - v^k_{i+\frac{1}{2}} \left(v^k_{i+1} - v^k_i\right)\right] \tag{3.19}$$

This very last way of expressing the term is the one unofficially called **semi-conservative** due to its appearance, which might be in TRUST papers.

3.3.4 Summary

Here are summed up the different expressions obtained for the momentum equation where $\mathcal{R}$ denotes the only changing part between possible schemes :

$$\Delta x_{i+\frac{1}{2}} \left[\alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}}\right]_c \frac{v^{k,n+1}_{i+\frac{1}{2}} - v^{k,n}_{i+\frac{1}{2}}}{\Delta t} + \mathcal{R} + \left[\alpha^k_{i+\frac{1}{2}}\right]_u \left(p^k_{i+1} - p^k_i\right) = 0 \tag{3.20}$$

with $\mathcal{R}$ chosen within relations derived at 3.17, 3.18 or 3.19 :

$$\mathcal{R} = \begin{cases} \left[\alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}}\right]_u v^k_{i+\frac{1}{2}} \left(\left[v^k_{i+1}\right]_u - \left[v^k_i\right]_u\right) \text{ or} \\ \frac{1}{2} \left[\alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}}\right]_u \left(\left[v^k_{i+1}\right]_u{}^2 - \left[v^k_i\right]_u{}^2\right) \text{ or} \\ \left[\alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}}\right]_u \left(\left[v^k_{i+1}\right]_u{}^2 - \left[v^k_i\right]_u{}^2 - v^k_{i+\frac{1}{2}} \left(\left[v^k_{i+1}\right]_u - \left[v^k_i\right]_u\right)\right) \end{cases} \tag{3.21}$$

Note that a differentiation between density on time and space derivative is made for interpolation. This is consecutive to the meeting with TRUST member A. Gerschenfeld who explained that this trick is used in the plateform.

3.4 Conservative momentum equation discretization

The previous choice of discretization for the momentum equation was guided but the idea of getting closer of what is implemented in TRUST . However, it seemed interesting to also be able to preserve the conservativity of the momentum equation (for mono-fluid case at least), which is one of its intrinsic properties :

$$\frac{\partial(\alpha^k \rho^k v^k)}{\partial t} + \frac{\partial(\alpha^k \rho^k v^k v^k)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{3.22}$$

Here we do not reformulate the equation but directly apply finite volumes, integrating on the **dual mesh** cells as velocity is defined at interfaces because of the use of a staggered scheme.

3.4.1 Time discretization

As always, using Euler forward explicit method for time integration provides :

$$\int_{c_{i+\frac{1}{2}}} \frac{\partial(\alpha^k \rho^k v^k)}{\partial t} dx = \Delta x_{i+\frac{1}{2}} \frac{\alpha^{k,n+1}_{i+\frac{1}{2}} \rho^{k,n+1}_{i+\frac{1}{2}} v^{k,n+1}_{i+\frac{1}{2}} - \alpha^{k,n}_{i+\frac{1}{2}} \rho^{k,n}_{i+\frac{1}{2}} v^{k,n}_{i+\frac{1}{2}}}{\Delta t} \tag{3.23}$$

Beware, k still denotes the specie, $i+\frac{1}{2}$ concerns the spatial position of the value and exponent n concerns the time step at which it is taken.

3.4.2 Space discretization

This time there is a single scheme to derive although various interpolation method could be chosen. Integrating momentum advection term over a dual mesh's cell gives :

$$\int_{c_{i+\frac{1}{2}}} \frac{\partial(\alpha^k \rho^k v^k v^k)}{\partial x} dx = \alpha^k_{i+1} \rho^k_{i+1} v^k_{i+1} v^k_{i+1} - \alpha^k_i \rho^k_i v^k_i v^k_i \tag{3.24}$$

and on the pressure gradient :

$$\int_{c_{i+\frac{1}{2}}} \alpha^k \frac{\partial(p^k)}{\partial x} dx = \alpha^k_{i+\frac{1}{2}} \left(p_{i+1} - p_i\right) \tag{3.25}$$

which is the same as in the non conservative momentum equation formulation.

3.4.3 Summary

When all discretized terms are put together, the conservative momentum equation scheme is :

$$\begin{aligned} &\Delta x_{i+\frac{1}{2}} \frac{\left[\alpha^{k,n+1}_{i+\frac{1}{2}} \rho^{k,n+1}_{i+\frac{1}{2}}\right]_c v^{k,n+1}_{i+\frac{1}{2}} - \left[\alpha^{k,n}_{i+\frac{1}{2}} \rho^{k,n}_{i+\frac{1}{2}}\right]_c v^{k,n}_{i+\frac{1}{2}}}{\Delta t} \\ &+ \alpha^k_{i+1} \rho^k_{i+1} \left[v^k_{i+1}\right]_u \left[v^k_{i+1}\right]_u - \alpha^k_i \rho^k_i \left[v^k_i\right]_u \left[v^k_i\right]_u \\ &+ \left[\alpha^k_{i+\frac{1}{2}}\right]_u \left(p_{i+1} - p_i\right) = 0 \end{aligned} \tag{3.26}$$

where interpolation is performed for colored terms, fully upwind.

3.5 Total energy equation discretization

In order to have a fully conservative scheme, which should have better performances than all the others, we provide discretization of the total energy equation, which we recall :

$$\frac{\partial(\alpha^k \rho^k e^k)}{\partial t} + \frac{\partial(\alpha^k \rho^k e^k v^k)}{\partial x} + \frac{\partial(\alpha^k p^k v^k)}{\partial x} + p^k \frac{\partial(\alpha^k)}{\partial t} = 0 \tag{3.27}$$

We choose to define total energy at cells centers, on the primal mesh. Thus we will perform the integration on an arbitrary cell c_i.

3.5.1 Time discretization

Using explicit Euler forward time integration brings to :

$$\int_{c_i} \frac{\partial(\alpha^k \rho^k e^k)}{\partial t} dx = \Delta x_i \frac{\alpha^{k,n+1}_i \rho^{k,n+1}_i e^{k,n+1}_i - \alpha^{k,n}_i \rho^{k,n}_i e^{k,n}_i}{\Delta t} \tag{3.28}$$

$$\int_{c_i} p^k \frac{\partial(\alpha^k)}{\partial t} dx = \Delta x_i p_i \frac{\alpha^{k,n+1}_i - \alpha^{k,n}_i}{\Delta t} \tag{3.29}$$

where no specific interpolation is required as all variables are defined at cells centers.

3.5.2 Space discretization

Discretizing the total energy convection term on the primal mesh provides

$$\int_{c_i} \frac{\partial(\alpha^k \rho^k e^k v^k)}{\partial x} dx = \alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}} e^k_{i+\frac{1}{2}} v^k_{i+\frac{1}{2}} - \alpha^k_{i-\frac{1}{2}} \rho^k_{i-\frac{1}{2}} e^k_{i-\frac{1}{2}} v^k_{i-\frac{1}{2}} \tag{3.30}$$

and the pressure physical work is derived as

$$\int_{c_i} \frac{\partial(\alpha^k p^k v^k)}{\partial x} dx = \alpha^k_{i+\frac{1}{2}} p^k_{i+\frac{1}{2}} v^k_{i+\frac{1}{2}} - \alpha^k_{i-\frac{1}{2}} p^k_{i-\frac{1}{2}} v^k_{i-\frac{1}{2}} \tag{3.31}$$

Note that physical work has very different nature from convection as it is responsible for kinetic energy variations, meaning in our case for internal energy too (as kinetic energy can be converted into internal energy). Because of this nature difference, it could be interesting to push investigations for interpolations over this term and address these differences.

3.5.3 Summary

The total energy equation scheme is then formulated as

$$\begin{aligned} &\Delta x_i \frac{\alpha_i^{k,n+1} \rho_i^{k,n+1} e_i^{k,n+1} - \alpha_i^{k,n} \rho_i^{k,n} e_i^{k,n}}{\Delta t} \\ &+ \left[\alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}} e^k_{i+\frac{1}{2}}\right]_u v^k_{i+\frac{1}{2}} - \left[\alpha^k_{i-\frac{1}{2}} \rho^k_{i-\frac{1}{2}} e^k_{i-\frac{1}{2}}\right]_u v^k_{i-\frac{1}{2}} \\ &+ \left[\alpha^k_{i+\frac{1}{2}}\right]_u \left[p^k_{i+\frac{1}{2}}\right]_c v^k_{i+\frac{1}{2}} - \left[\alpha^k_{i-\frac{1}{2}}\right]_u \left[p^k_{i-\frac{1}{2}}\right]_c v^k_{i-\frac{1}{2}} + \Delta x_i p_i \frac{\alpha_i^{k,n+1} - \alpha_i^{k,n}}{\Delta t} = 0 \end{aligned} \tag{3.32}$$

Beware, terms of different color are interpolated separately whereas gathered colored terms are interpolated as conserved quantities.

3.6 VDF scheme

During this internship, it was not clear what `TRUST` really uses for the momentum equation. As `TRUST` is inspired from American `TRACE` code[18], this is the scheme that is the most likely to be implemented, but nothing is really sure as no genuine documentation exists.

Density and internal energy equations are preserved and identical as the one we derived previously. However, momentum equation is expressed in a different way, keeping whole momentum terms in advection term and expressing it under the so-called *semi-conservative* form :

$$\frac{\partial(\alpha^k \rho^k v^k)}{\partial t} + \frac{\partial(\alpha^k \rho^k v^k v^k)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{3.33}$$

$$\iff \alpha^k \rho^k \frac{\partial(v^k)}{\partial t} + v^k \frac{\partial(\alpha^k \rho^k)}{\partial t} + \alpha^k \rho^k v^k \frac{\partial(v^k)}{\partial x} + v^k \frac{\partial(\alpha^k \rho^k v^k)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{3.34}$$

$$\iff \alpha^k \rho^k \frac{\partial(v^k)}{\partial t} - v^k \frac{\partial(\alpha^k \rho^k v^k)}{\partial x} + \alpha^k \rho^k v^k \frac{\partial(v^k)}{\partial x} + v^k \frac{\partial(\alpha^k \rho^k v^k)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{3.35}$$

$$\iff \alpha^k \rho^k \frac{\partial(v^k)}{\partial t} + \frac{\partial(\alpha^k \rho^k v^k v^k)}{\partial x} - v^k \frac{\partial(\alpha^k \rho^k v^k)}{\partial x} + \alpha^k \frac{\partial(p^k)}{\partial x} = 0 \tag{3.36}$$

To go from 3.33 to 3.34, derivatives have been expanded. From 3.34 to 3.35 density equation is used to convert density time evolution term into a density advection one. Finally for 3.35 to 3.36 we gather two terms before pressure gradient to form a momentum advection term, leading to 3.36.

3.6.1 Time discetization

Using Euler forward first order method on the **dual mesh** provides

$$\int_{c_{i+\frac{1}{2}}} \alpha^k \rho^k \frac{\partial(v^k)}{\partial t} dx = \Delta x_{i+\frac{1}{2}} \alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}} \frac{v^{k,n+1}_{i+\frac{1}{2}} - v^{k,n}_{i+\frac{1}{2}}}{\Delta t} \tag{3.37}$$

where interpolation will have to be performed for density and volumic fraction.

3.6.2 Space discretization

First discretizing the momentum advection term gives

$$\int_{c_{i+\frac{1}{2}}} \frac{\partial(\alpha^k \rho^k v^k v^k)}{\partial x} dx = \alpha^k_{i+1} \rho^k_{i+1} v^k_{i+1} v^k_{i+1} - \alpha^k_i \rho^k_i v^k_i v^k_i \tag{3.38}$$

for the second part of the semi-conservative term

$$\int_{c_{i+\frac{1}{2}}} v^k \frac{\partial(\alpha^k \rho^k v^k)}{\partial x} dx = v^k_{i+\frac{1}{2}} \left(\alpha^k_{i+1} \rho^k_{i+1} v^k_{i+1} - \alpha^k_i \rho^k_i v^k_i \right) \tag{3.39}$$

and finally for the pressure gradient term we get

$$\int_{c_{i+\frac{1}{2}}} \alpha^k \frac{\partial(p^k)}{\partial x} dx = \alpha^k_{i+\frac{1}{2}} \left(p^k_{i+1} - p^k_i \right) \tag{3.40}$$

3.6.3 Summary

To sum up VDF scheme, recall that density and internal energy equations are the same as what has been derived before. Only the momentum equation changes, which scheme is

$$\begin{aligned} &\Delta x_{i+\frac{1}{2}} \left[\alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}} \right]_c \frac{v^{k,n+1}_{i+\frac{1}{2}} - v^{k,n}_{i+\frac{1}{2}}}{\Delta t} \\ &+ \left(\alpha^k_{i+1} \rho^k_{i+1} \left[v^k_{i+1} \right]_u \left[v^k_{i+1} \right]_u - \alpha^k_i \rho^k_i \left[v^k_i \right]_u \left[v^k_i \right]_u \right) - v^k_{i+\frac{1}{2}} \left(\alpha^k_{i+1} \rho^k_{i+1} \left[v^k_{i+1} \right]_u - \alpha^k_i \rho^k_i \left[v^k_i \right]_u \right) \\ &+ \left[\alpha^k_{i+\frac{1}{2}} \right]_u \left(p^k_{i+1} - p^k_i \right) = 0 \end{aligned} \tag{3.41}$$

where on the contrary to our previous semi-conservative scheme, the density is **inside** space derivatives. One advantage of this method is that interpolation mainly has to be performed on the velocity. On the previous semi-conservative scheme, interpolation had to be done on lots of other variables. As said previously, this discretization has the asset to have a good stability[17, 18] while having a second order pressure gradient.

3.7 Antoine LLOR's scheme

This scheme was designed by Antoine Llor, researcher at the CEA, who I have had the occasion to speak with about my work in this internship. Working on staggered schemes for a long time, he derived a scheme for us that is fully conservative while being under internal energy formulation. This is done writing the kinetic energy equation looking for diffusion terms i.e. kinetic energy converted into internal energy. Adding this dissipated energy to internal energy equation ensures total energy conservation. This principle has been presented in 1974 by Debar[19], and then studied by others such as Vasquez and Llor[20]. More details about this scheme are available in appendix E. The proof of its conservativity can be found in appendix F.

3.8 Second order accuracy in space

In order to get even better performance out of our schemes we perform a paradigm shift on the interpolation method. Rather than computing weighted sums of neighbouring values for first order, a gradient term will be added to the Taylor development on a cell c_i :

$$\Phi = \Phi_i + (\nabla\Phi)_i(x - x_i) \tag{3.42}$$

In other words, scalar functions of interest will be considered piece-wise affine over the domain and continuous on each cells. Note that all variables considered here are taken at **same time step** n.

3.8.1 Interpolation for node values

Consider Φ a scalar variable defined at cells centers which we need at interface $i + \frac{1}{2}$. To get this value, we must consider both sides of the interface to know possible values :

- $\Phi^-_{i+\frac{1}{2}} = \Phi_i + \frac{\Delta x_i}{2} \left.\frac{\partial(\Phi)}{\partial x}\right|_i + \mathcal{O}(\frac{\Delta x_i^2}{4})$ (left)
- $\Phi^+_{i+\frac{1}{2}} = \Phi_{i+1} - \frac{\Delta x_{i+1}}{2} \left.\frac{\partial(\Phi)}{\partial x}\right|_{i+1} + \mathcal{O}(\frac{\Delta x_{i+1}^2}{4})$ (right)

where only half cell weight is taken into account as the aim is to interpolate on the dual mesh.

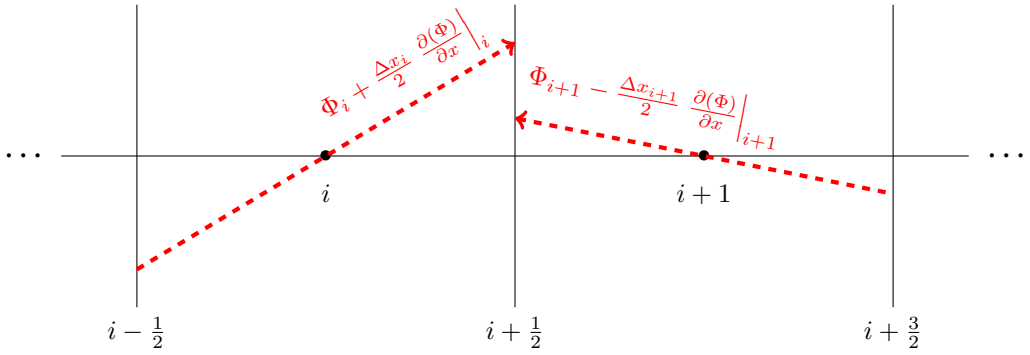


Figure 3.2: Illustration of the underlying working principle of second order approximation for a staggered scheme, here to interpolate interface values knowing node ones.

Then depending on the reader's will, many possibilities are offered to choose $\Phi_{i+\frac{1}{2}}$ value such as

$$\Phi_{i+\frac{1}{2}} = \begin{cases} \Phi^-_{i+\frac{1}{2}} \text{ if } v_{i+\frac{1}{2}} > 0 \\ \Phi^+_{i+\frac{1}{2}} \text{ if } v_{i+\frac{1}{2}} < 0 \end{cases} \quad \text{(upwind)} \tag{3.43}$$

$$\Phi_{i+\frac{1}{2}} = \frac{1}{2}(\Phi_i + \Phi_{i+1}) \quad \text{(centered)} \tag{3.44}$$

Term $\left.\frac{\partial(\Phi)}{\partial x}\right|_i$ has to be defined. This will be done using limiters, which working principle will be explained further in this chapter. Limiter function will be denoted $f : \mathbb{R}^2 \to \mathbb{R}$ and provide this expression for the gradient on cell i :

$$\left.\frac{\partial(\Phi)}{\partial x}\right|_i = f\left(\left.\frac{\partial(\Phi)}{\partial x}\right|_{i-\frac{1}{2}}, \left.\frac{\partial(\Phi)}{\partial x}\right|_{i+\frac{1}{2}}\right) \tag{3.45}$$

where gradients given to f are computed with first order finite differences schemes :

$$\left.\frac{\partial(\Phi)}{\partial x}\right|_{i+\frac{1}{2}} = \frac{\Phi_{i+1} - \Phi_i}{\Delta x_{i+\frac{1}{2}}} \tag{3.46}$$

To summarize the method, we end up with these values for $\Phi^-_{i+\frac{1}{2}}$ and $\Phi^+_{i+\frac{1}{2}}$:

$$\Phi^-_{i+\frac{1}{2}} = \Phi_i + \frac{\Delta x_i}{2} f\left(\left.\frac{\partial(\Phi)}{\partial x}\right|_{i-\frac{1}{2}}, \left.\frac{\partial(\Phi)}{\partial x}\right|_{i+\frac{1}{2}} \right) \tag{3.47}$$

$$\Phi^+_{i+\frac{1}{2}} = \Phi_{i+1} - \frac{\Delta x_{i+1}}{2} f\left(\left.\frac{\partial(\Phi)}{\partial x}\right|_{i+\frac{1}{2}}, \left.\frac{\partial(\Phi)}{\partial x}\right|_{i+\frac{3}{2}} \right) \tag{3.48}$$

If the need of interpolation arises for interface $c_{i-\frac{1}{2}}$ for example, a shift of the stencil to the left enables to find the method to use.

3.8.2 Interpolation for cells centers values

The same approach is applied while shifting the stencil from the dual mesh to the primal one. Consider v which is defined at interfaces and which we look to interpolate at cells centers :

- $v_i^- = v_{i-\frac{1}{2}} + \frac{\Delta x_{i-\frac{1}{2}}}{2} \left.\frac{\partial(v)}{\partial x}\right|_{i-\frac{1}{2}} + \mathcal{O}(\frac{\Delta x^2_{i-\frac{1}{2}}}{4})$ (left)
- $v_i^+ = v_{i+\frac{1}{2}} - \frac{\Delta x_{i+\frac{1}{2}}}{2} \left.\frac{\partial(v)}{\partial x}\right|_{i+\frac{1}{2}} + \mathcal{O}(\frac{\Delta x^2_{i+\frac{1}{2}}}{4})$ (right)

For instance, v_i can be interpolated as

$$v_i = \begin{cases} v_i^- \text{ if } v_{i-\frac{1}{2}} + v_{i+\frac{1}{2}} \geq 0 \\ v_i^+ \text{ if } v_{i+\frac{1}{2}} + v_{i-\frac{1}{2}} < 0 \end{cases} \quad \text{(upwind)} \tag{3.49}$$

$$v_i = \frac{1}{2}\left(v_{i-\frac{1}{2}} + v_{i+\frac{1}{2}} \right) \quad \text{(centered)} \tag{3.50}$$

The notation for the velocity gradient at interfaces $i+\frac{1}{2}$ shall be defined in a similar way as before. In fact, considering a limiter function f, which enables to reduce oscillations, we obtain :

$$\left.\frac{\partial(v)}{\partial x}\right|_{i+\frac{1}{2}} = f\left(\left.\frac{\partial(v)}{\partial x}\right|_{i}, \left.\frac{\partial(v)}{\partial x}\right|_{i+1} \right) \tag{3.51}$$

with $\left.\frac{\partial(v)}{\partial x}\right|_i$ being computed using a first order finite difference scheme :

$$\left.\frac{\partial(v)}{\partial x}\right|_{i} = \frac{v_{i+\frac{1}{2}} - v_{i-\frac{1}{2}}}{\Delta x} \tag{3.52}$$

To summarize, we end up with these values for v_i^- and v_i^+ :

$$v^-_i = v_{i-\frac{1}{2}} + \frac{\Delta x_{i-\frac{1}{2}}}{2} f\left(\left.\frac{\partial(v)}{\partial x}\right|_{i-1}, \left.\frac{\partial(v)}{\partial x}\right|_{i} \right) \tag{3.53}$$

$$v^+_i = v_{i+\frac{1}{2}} - \frac{\Delta x_{i+\frac{1}{2}}}{2} f\left(\left.\frac{\partial(v)}{\partial x}\right|_{i}, \left.\frac{\partial(\Phi)}{\partial x}\right|_{i+1} \right) \tag{3.54}$$

Note that the term $v_{i+\frac{3}{2}}$ will be involved to get $\left.\frac{\partial(\Phi)}{\partial x}\right|_{i+1}$ in v_i^+ value computation, same with $v_{i-\frac{3}{2}}$ when computing $\left.\frac{\partial(\Phi)}{\partial x}\right|_{i-1}$. Stencil's footprint is then one term larger as before, being consistent with a second order approach. As before, a shift of the stencil then enables to find interpolation method for other centers such as c_{i+1}. For values which are just after or before the boundary, we perform first order approximation.

3.8.3 Examples of limiters

For simplicity sake, we will keep the number of limiters small and we will stick with the most common ones. Consider $a, b \in \mathbb{R}$, then a few common limiters are :

$$\text{minmod}(a,b) = \max(0, \min(a,b)) + \min(0, \max(a,b)) \tag{3.55}$$

$$\text{superbee}(a,b) = \max(0, \min(2a,b), \min(a,2b)) \tag{3.56}$$

$$\text{umist}(a,b) = \max\left[0, \min(2a, \frac{a}{4} + \frac{3b}{4}, \frac{3a}{4} + \frac{b}{4}, 2b)\right] \tag{3.57}$$

As M. Bijan[21] explains, a limiter interest resides in its ability to select within $0, a$ or b the gradient value that will avoid creating new extrema i.e. staying in the so called *Total Variation Diminishing* (TVD) region. As it does not create new extremas, no oscillation can arise from this method, meaning the approximation will get better without becoming oscillatory.

3.9 Overview of the available schemes and notations

This makes a lot of schemes to investigate. Recall that two paradigms exist : working in specific internal energy or specific total energy formulation. This will induce different conservativity properties on the schemes, meaning different accuracies thus efficient or not shock capture. Same idea can be developed for momentum. **Note that all presented schemes a conservative for the density.**

All available schemes are summed up in the following table. Energy paradigm is to be read column-wise whereas momentum conservativity should be read row-wise. The momentum advection term shape is recalled on last columns. Colours are meant to emphase conservativity of the scheme. Colour **red** is used for schemes preserving total energy, basically any that is under total energy formulation. Colour **blue** for momentum conservative schemes, and colour **green** for scheme that conserve both momentum and total energy.

		Specific energy		
		Internal	**Total**	Momentum alternative
Momentum	**Non-conservative**	MacNCVε	MacNCVe	$v\frac{\partial(v)}{\partial x}$
		MacSCVε	MacSCVe	$\frac{\partial(v^2)}{\partial x} - v\frac{\partial(v)}{\partial x}$
		MacQCVε	MacQCVe	$\frac{1}{2}\frac{\partial(v^2)}{\partial x}$
		MacVDFε	MacVDFe	$\frac{\partial(\alpha\rho v^2)}{\partial x} - v\frac{\partial(\alpha\rho v)}{\partial x}$
	Conservative	MacCMε	MacCMe	
		MacALLOR		

Table 3.1: Notation to mention schemes depending on their momentum conservativity and their energy formulation

Last letter stands for the type of energy formulation which is used (ε : internal, e : total). NCV stands for *non-conservative* form for the momentum equation, SCV stands for *semi-conservative term*, which is not mathematically rigorous but is intuitive considering the visual aspect of the corresponding term. Finally, QCV stands for *quasi-conservative* as it looks like a conservative term but is not a real one, and CM for *conservative momentum*. ALLOR and VDF stand for Antoine Llor's scheme and `TRUST` VDF scheme.

Chapter 4

Presentation of the code

As the whole internship is based on the development of a code needed for generating approximations to problems, we will present its overall structure, working principles and underlying concepts.

4.1 Architecture of the project

To project is articulated around configuration files (`.json`), some `Python` pre and post-processing and a `C++` numerical core.

4.1.1 Tree structure of files

In the following development, directory `/hydromockup` will be called *root* directory as it is the root of the project. At the moment of this report being written, directories branching is the following :

```
+/hydromockup
|   +-/build
|   +-/doc
|   +-/References
|   +-/Results
|   +-/src
|   +-/studies
|   +-/tests
|   +-/Validation
+- create_project(.sh)
+- hydromockup(.py)
+- plot(.py)
+- README.md
```

The aim of this file structure is to have a clean organization enabling easy and secure tests execution in order not to break the program. It is a way to tend to performance, readability and integrity of the code. Here are presented the role of each directory or file located at the root.

Directories

- `/build` : Hosts `cmake` files for compilation (except `CMakeLists.txt`) and will host binaries after compilation, for instance object files.
- `/doc` : The Doxyfile file for `doxygen` generated documentation as well as a `css` sheet for a cleaner looking `html` documentation are stored here. This directory will host generated documentation with `html` and LaTeX format.
- `/References` : Analytical solutions for test cases are stored here. Has no sub branching.

- `/Results` : Stores generated `json` result files after running a simulation. It is then empty when cloning the project. No inner branching.
- `/src` : Contains the `C++` sources needed for the numerical core to run (`.h`, `.cc` files) as well as `Eigen` and `rapidjson` libraries. `CMakeLists.txt` can be found here.
- `/studies` : Playground of the project, where personal scripts can be put to run the code without breaking the global architecture.
- `/tests` : Storage for test cases, `json` files with pre-defined values. For example, sedimentation problems, Sod shock tube and others.
- `/Validation` : Not used during this internship. Will be filled with test cases in order to perform program benchmarks.

Root files

Note that as these file do not have an extension (`.py` or `.sh`) they can be executed straight-forward in the console by entering : `./name_of_file` from the root.

- `create_project` : `bash` code that can be hand-used to compile the project and generate the doxygen documentation.
- `hydromockup` : `Python` code that performs the pre-processing and launches the computational kernel. It provides a git commit ID to each simulation, stores files on the temporary memory, calls `cmake` compiler and writes output data in a `json` file at the end of the computations. In a normal use, this script does not have to be modified but **it should always be called to execute this entire program**.
- `plot` : `Python` script to perform post-processing. It should be called by hand and has many options to animate plots of change their scale. No mandatory to use.
- `README.md` : only for Git presentation page. Brief description of the project.

4.1.2 Numerical core source files

The choice of implementing such a code with a `C++` core was done taking into account that some parts of it already existed, that this is an efficient language when it comes to large amounts of computation and of course, it is a language enabling object oriented programming and inheritance.

- `bc_type.h` : declares a type `BC_type` for available boundary conditions types such as Dirichlet or periodic ones. It does not define the values container.Boundary condition `None` corresponds to letting boundary value constant in time, they are not modified.
- `datastructure.h` : declares a class `Data` that will have almost all attributes and methods in a public scope in order for the other objects to access them. This data structure will only be modified when initialized at the beginning of simulation, and when performing time integration. Otherwise, other class can only **access** its elements, not over-write them. A variety of other objects are declared such as `LimiterType`, `BoundaryCondition` where a container for values is available, `FaceProjection` for the available ways of interpolating values.
- `eos.h` : hosts equation of state related methods, for now about stiffened gas one. Enables to deduce pressure and sound speed. Code exits if sound speed is infinite.
- `error.h` : customized function to have clear error logs in the command invite. Can be imported in any other file.
- `limiter.h` : declares a type for limiters, used for second order only. Some of them are *minmod, superbee, umist* for example.

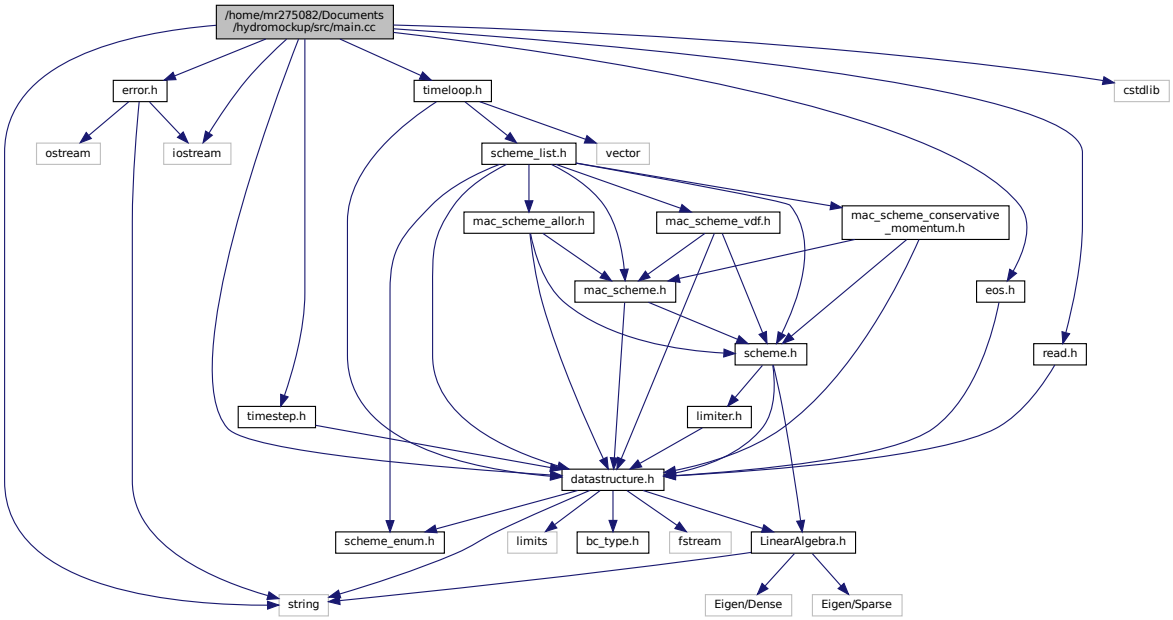


Figure 4.1: File branching of the numerical core, generated with doxygen. In dark gray is the main program, then in black outlined boxes are the different files needed by the main program, and in light gray outlined boxes are the external libraries required.

- `LinearAlgebra.h` : declares customized names for `Eigen` vector or matrices to avoid long declaration with all template arguments. Also declares printing methods for such objects.
- `mac_scheme_allor.h` : implementation of Antoine Llor's scheme.
- `mac_scheme_conservative_momentum.h` : implementation of the momentum conservative formulation of Mac scheme.
- `mac_scheme_vdf.h` : implementation of what is supposed to be `TRUST` VDF scheme.
- `mac_scheme.h` : implementation of the basic version of Mac scheme.
- `read.h` : set of methods for data structure filling. Error management on parameter values can be done here. It initializes all need variables for the program coming from the `json` configuration file i.e. test case file.
- `scheme_enum.h` : defines enumerated type for schemes for in-code scheme management.
- `scheme_list.h` : memory allocator set of methods to avoid stack overfilling.
- `scheme.h` : declaration of abstract class to be used as root for staggered or not schemes. Not supposed to be used but can be changed if a need to create new method common to all schemes.
- `timeloop.h` : declares methods to perform time integration of variables depending on the momentum conservativity property of the schemes. This might be the file to change if starting to implement implicit or split time methods.
- `timestep.h` : hosts time step computing methods.

4.1.3 Compilation and documentation

Compile the code

In order to compile the code, different options are available. The first one is to execute by hand all the required commands, starting at the root of the project :

bash

```
cd build
cmake ../src
make -j8
```

an other possibility is to use the provided script `create_project` located at the root, that performs these commands itself and additionally creates the doxygen documentation. Finally, if using the provided pre-processing code `hydromockup`, the compilation is made by this file (through an `os.command()` line), meaning there is no need to compile the project by hand.

Note that compilation is made using `cmake` as it simplify a lot compilation instructions. The `CMakeLists.txt` file is placed in the `/src` directory and the other compilation files are placed in `/build` directory.

Documentation generation

The documentation is made using `doxygen` as it is an open library which is simple to use and to compile. In order to generate the documentation, the following commands have to be entered, from the root :

bash

```
cd ./doc
doxygen Doxyfile
cd ./doxygen_doc/latex
pdflatex -interaction=batchmode refman.tex
```

The two last lines aim at compiling the LaTeXdocumentation to generate the PDF version. Regarding the interactions with the architecture, two new directories appear in `/doc/doxygen_doc` which are `/html` and `/latex` where `doxygen` generates the documentation under the corresponding format. An other possibility to get the documentation to be generated is to use `create_project` located at the root.

To find the {*format*} documentation generated by Doxygen, find one of these files:

- `html` : `/doc/doxygen_doc/html/index.html`
- LaTeX: `/doc/doxygen_doc/latex/refman.pdf`

4.2 Code operation

4.2.1 Important classes

The code is based around object oriented programming, however for practical reasons, all attributes are public in order to be accessible for other objects.

`Data` class

The central object of this program is a big container for all needed data such as matrices for variables, thermodynamic values, simulations parameters such as time step or artificial viscosity, as well

as options to make schemes change.

This data structure is initialized in the main program according to entries of the configuration file. No other class can write on its attributes other than `Timeloop` class, which performs time integration and need to update variable matrices. This means that the `Data` class is only passed by reference and even a `const` reference. This avoids making copies of it at each time step, thus increasing performance. The `Data` class does not aim at computing or performing any time or spatial integration. It is only used to store data in-code and to write it to files when needed.

`Scheme` class

The other important class of the program is the `Scheme` class which is the one from which all of our schemes inherits. It is an abstract class that enables to differentiate between staggered or co-localized schemes. Its purpose is to structure the code, not to be used.

For instance, `MacScheme` is the first which inherits from it as it is the most basic form of the Mark-and-Cell scheme. All other schemes that have been studied in this internship then inherits from `MacScheme` as they are all staggered with velocity at interfaces.

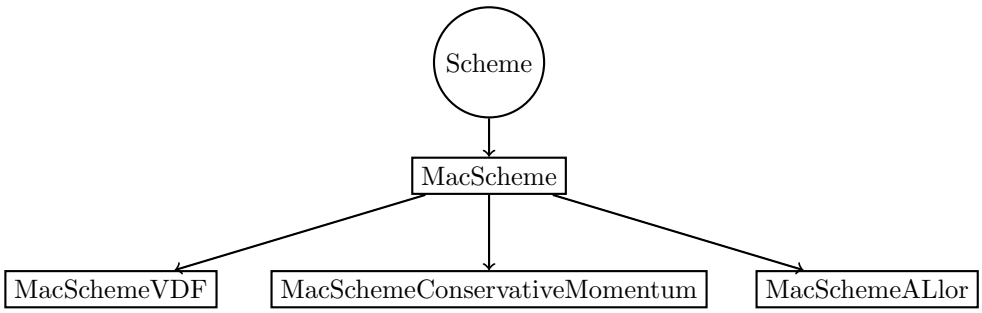


Figure 4.2: Inheritance illustration in the available schemes of the project.

Having this kind of inheritance structure enables to extend the code easily to other staggered schemes or even to consider other types of variable mesh shifts.

4.2.2 General information

It is now time to see how the code can be used. The overall idea is that the code will always take a configuration file (`json` format), fill the in-code data structure, perform all needed computations, and then write back the results with potentially asked intermediary data into a file.

Configuration file

All configuration files contain same arguments. Note that the `@help` ones are not meant to be used as argument but are used as comments for `json` file. All these arguments have filtered values, and **the code is made so that it stops if an unexpected value occurs** rather than correcting it. An example of such a configuration file can be found in appendix G. **Mesh parameters are mandatory**, meaning the code will crash if a single one is missing, but others can be missing ; the code might have a strange behaviour. Note that the values of those parameters, for some of them are stored in lists in order to enable the conversion to multi-fluid problems more easily.

- `case_name` : It can be used to implement test cases or custom initializations directly in the code (at the very end of file `read.cc`).
- `mesh["Xmin"], mesh["Xmax"]` : Real number for start and end of the segment. If $x_{\min} \geq x_{\max}$ then the code exits with an error message.

- `mesh["N"], mesh["nb_mat"]` : Number of cells in the domain ; should be a positive integer. Code crashes otherwise.

- `scheme` : name of the scheme, string ; accepted values are `Mac, Mac_CM, MacVDF, MacALLOR`. Any case, with or without underscore is accepted. `Mac_Conservative_Momentum` is also accepted instead of `Mac_CM`, again with any mix of upper and lower case, with or without some of those underscores. Code exits if non valid entry is provided.

- `linear_artificial_viscosity_coefficient` : Real numbers, code crashes otherwise.

- `quadratic_artificial_viscosity_coefficient` : Real numbers, code crashes otherwise.

- `face_projection, face_projection_for_momentum` : string, projection type used for variables. The projection for momentum concerns α and ρ inside momentum equation. Accepted values are `upwind, centered`, with any mix of upper and lower case, code exits if provided value is not within these ones.

- `dt` : time step, real number. If positive, the code assumes the user wants a constant time step for the simulation, that will also be the initial time step. If negative, the code will assume the user wants a dynamic time step. If null, code exits.

- `v_grad_v_term_type` : string for the type of expression for operator $v\frac{\partial(v)}{\partial x}$ in momentum equation under non conservative Mac scheme. Accepted values are `ncv, scv, qcv` respectively for non conservative, semi-conservative and quasi-conservative. Code exits if value is not managed. Note that these names have been chosen according to the look of the mathematical term they refer to, but not to any checked or proved conservativity property.

- `mixed_projection_alpha_rho_for_momentum` : boolean `true` or `false`. Used to force time derivative related density in momentum equation to be interpolated according to `face_projection`, and the one on space derivative to be upwind. Code crashes if value is not a `json` boolean. This option only affects MacNCV, MacQCV and Mac SCV schemes. If `false`, velocity in momentum equation will be interpolated according to `face_projection` and volumic fraction and density in momentum equation will be interpolated according to `face_projection_for_momentum`. If `true`, the same holds, but volumic fraction and density related to space derivative are forced to be interpolated `upwind`.

- `total_energy_formulation` : boolean `true` or `false`. Used to choose to work under the total energy formulation of Euler equations (`true`) or under specific internal energy (`false`). Code crashes if value is not a `json` boolean.

- `pressure_projection_for_total_energy` : string, way of interpolating pressure gradient when working under total energy formulation. Beware, this argument is still required when working under specific internal energy, to be initialized in data structure. However it will not be used. Accepted values are : `upwind, centered, downwind`, with any mix of upper and lower case, code exits if provided value is not within these ones.

- `flux_limiter` : string, accepted values are `minmod, umist, superbee` with any mix of upper and lower case. It designates the flux limiter used in case of order 2. This should be always initialized, even when working at order 1. However it will not be used. Code exits if limiter name is not managed.

- `appx_order` : 1 or 2. Otherwise the code exits if it is an other integer. The code crashes if the type is an other one. It represents the wanted order of approximation (first or second).

- `xSeparation` : real number for the x-coordinate state change in the Riemann problem. If smaller than $x_{\min}$ it will be considered that the right state of the Riemann problem is the one for the whole domain. Similarly, if bigger than $x_{\max}$, the code assumes the left state of the Riemann problem is the one for the whole space. If the type is not respected, the code crashes.

- `alpha_L, alpha_R` : Left and right volumic fraction for the Riemann problem. Code exits if both sides of Riemann problem do not sum to 1 (barycentric condition). Code crashes if `double` type is not respected.
- `rho_L, rho_R` : Left and right density for the Riemann problem. Should be a real positive number. Code exits if not positive. Code crashes if type is not corresponding.
- `eps_L, eps_R` : Left and right internal specific energy for the Riemann problem. Should be a real positive number. Code exits if not positive. Code crashes if type is not corresponding.
- `u_L, u_R` : Left and right velocity for the Riemann problem. Should be a real number. Code crashes if type is not corresponding.
- `gamma` : adiabatic coefficient of the fluid. Code crashes if `double` type is not respected.
- `Pi` : constant representing molecular attraction in the stiffened gas equation (p^* or p_∞). Code crashes if `double` type is not respected.
- `gravity_source_term` : gravity attraction (only acts in momentum and total energy equations). Code crashes if `double` type is not respected.
- `final_time` : real number corresponding to dimensionless time at which simulation will stop. No filtering is performed, code will crash if negative or non real value.
- `cfl` : real coefficient, CFL coefficient used for time integration. No filtering is performed, code will crash if value has a wrong type.
- `leftBC, rightBC` : boundary condition for the problem, string. Only accepted values are 'Dirichlet' and `None`. The `None` condition repeats the initial boundary values over and over (no time integration for first and last vector value). Code exits if the value is unknown.
- `leftBCValue, rightBCValue` : Real value for boundary condition. Should always be provided, even under `None` condition. Their vector form is to store $\alpha, \rho, v, \varepsilon$ in this order. Code crashes if type is not suiting.
- `output_file` : String, path and name for the file where results during and after the simulation should be written. No check of path validity is made. Code will crash if the path is wrong.
- `output_period` : real value. Results will be written after each `output_period` has passed in the inner simulation time frame. For instance, with simulation of dimensionless time 0.2, and `output_period`= 0.05, there will be only 5 results writing the `output_file` which are : initial state, and $0.05 * 4 = 0.2$ so 4 other states during the simulation.
- `conservativity_filename` : string, path and name for the file that will host conservativity data of the simulation (integrals of variables over the domain, as functions of time). No check of path validity is made, so code will crash if a problem arises.

Remark 1 Values that are set in the configuration file should be considered as default values if the user does not change them in an other script.

Remark 2 The name of the parameters that you find in the configuration file are the options that can be entered in the command line when executing the program. See 4.2.3 for some example.

Main program

The main program is located in file `/src/main.cc` by convention, and hosts simulation's skeleton. Its role is to call all needed objects' constructor to initialize memory spaces such as `Data` and `Timeloop` class.

Before starting time integration, it makes the first dumping to files to write the initial state of the simulation, both under variables' profiles and conservativity point of views.

At each time step, it calls the equation of state to get an updated pressure function (which is obtained thanks to density and specific internal energy). The main program then calls the time-step computing functions if needed to make the simulation go forward and after calling the variables value computing methods (scheme), performs some dumping to files if required in order to track variables profiles.

Once the final time is reach, defined by the user with `final_time`, a last dumping is performed to have final state of the simulation stored.

Generated files

When running the program, two files are generated. The first one is the conservativity file which stores variables' spatial integral over the domain; it is a `txt` file. The other one is the `json` result file which stores simulation parameters from the configuration file, as well a git commit ID if needed, and some states during the simulation depending on the `output_frequency` the user chose. An example of such a result file can be found in appendix H.

4.2.3 Example of execution

For executing the program, the configuration (i.e. test case) file should always be provided. In the following example, we will use the one located in `/tests` which is `Toro1.json`.

Through the command line

The general construction of the command (which should still be called if using an external script) is the following one, assuming we call the program from the root directory :

bash

```
./hydromockup <path_to_test_file> [--<opt_name> <opt_value>]
```

where the brackets indicate that there could be many options put in the command line. Recall that the `opt_name` is actually the name of the option that can be found in the configuration file, meaning any option that figures in the configuration file can be modified with such an command. Here is a concrete example :

bash

```
./hydromockup ./tests/Toro1.json --N 800 --output_file ./Results/my_file.json
```

Here the Toro1 test case, which we will see in the result chapter, is called. It is asked by `--N 800` to change the predefined number of cells to 800, and it is asked to name the result file `my_file.json` and to place it in directory `/Results`. Now it could be great to get a plot of the simulation, which is possible thanks to the command :

bash

```
./plot <result_file_path> [quantity_to_plot] [--<opt_name> <opt_value>]
```

Beware, if no quantity, such as density or velocity is asked, the script will assume the user wants all possibles quantities to be displayed (coordinates, density, volumic fraction, internal energy, pressure, sound speed, entropy and velocity). To see available options, enter `-h` or `--help` and a display of all options will be made in the console.

Going back to our example, we wish to plot density and velocity profiles at time 0.1, which can be made accordingly to this command :

bash

```
./plot ./Results/my_file.json density velocity --time 0.1
```

Many options are available, to change axis labels, to save the plot or to change the scale. Enter the help option for this script to see them.

With an external script

In this case, the principle is exactly the same as for the inline program execution : the `hydromockup Python` script at the root should be called, with a path to a test case file and be followed by the options the user wants to change from the configuration file. Commands are the very same as for previous case, only difference is that the external script should build it :

bash

```
./hydromockup <path_to_test_file> [--<opt_name> <opt_value>]
```

and the example presented before still holds. The advantage is that, assuming the user works with a custom `Python` script, post-processing can be performed with user's routine instead of the `plot` script which is provided with the project.

Assume the user has the following `Python` script, located in `/studies` :

```
import matplotlib.pyplot as plt
import os
import json

scheme_list = ["Mac_CM", "Mac_VDF", "Mac"]
cases = []
for scheme in scheme_list:
    outputfile = "res_file_" + scheme + ".json"
    command = "../hydromockup ../tests/Toro1.json"
    # Change values of the config file
    command += "\\\n   --scheme " + scheme
    command += "\\\n   --linear_artificial_viscosity_coefficient 0"
    command += "\\\n   --quadratic_artificial_viscosity_coefficient 0"
    command += "\\\n   --face_projection " + "upwind"
    command += "\\\n   --face_projection_for_momentum " + "centered"
    command += "\\\n   --N 800"
    command += "\\\n   --mixed_projection_alpha_rho_for_momentum " + "true"
    command += "\\\n   --dt " + "1.e-5"
    command += "\\\n   --output_file " + outputfile
    # Store configuration for post_proc
    cases.append({"outputfile": outputfile, "title": scheme, "scheme": scheme})
    # Execute command
    os.system(command)

# Plot
p_legend = list()
t_legend = list()

# Exact solution at final time
ref = json.load(open("../References/Toro1.json", "r"))
p,= plt.plot(ref["Position"], ref["Density"], "k-", lw=2)
p_legend.append(p)
t_legend.append("Reference")

# Custom results
for case in cases:
    resfile = json.load(open(case["outputfile"], "r"))
    data = resfile["results"][-1]["Materials"]["0"]
```

```
    terminated_successfully = resfile["terminated_successfully"]

    if (terminated_successfully):
            p,= plt.plot(data["x"], data["density"])
            p_legend.append(p)
            t_legend.append(case["title"])
    plt.legend(p_legend, t_legend)

plt.title("Density profile")
plt.savefig("density_profile_plot.pdf", format="pdf", bbox_inches='tight')
```

On the first part it is possible to see the option changes ask by the user, which iterates over some available schemes. Note that the pre-processing script `hydromockup` is called. In the second part, results files are opened and data is plotted in order to have a figure where everything is gathered. The dictionary enables to keep track of the different cases that are computed ; it could be extended to host even more options.

4.2.4 Git deposit

At the time of the writing of this report, the project is located on a git deposit, internal to the LMAG. The name of the deposit is `DTN/Collaboratif/rc607753/HydroMockup`. Beware, my work has been pushed on the branch `foxbranch` to preserve Rémi Chauvin's code integrity.

Thus, the complete command to clone the project and get the root directory `hydromockup` presented earlier is :

bash

```
git clone -b foxbranch <git_deposit_url>
```

where the url of the deposit has to be inserted. This should build the architecture presented earlier. **Note that the project's location might change in a close future.** It is to be discussed with my tutors.

Chapter 5

Results

5.1 Introduction

5.1.1 Precision about figures

In this section are presented some interesting results obtained with the code, making some parameters change such as artificial viscosity, the writing of some terms or energy formulation. The goal is to study schemes performance and behaviour when facing available parameter changes.

		Specific energy		
		Internal	**Total**	Momentum alternative
Momentum	**Non-conservative**	MacNCVε	MacNCVe	$v\frac{\partial(v)}{\partial x}$
		MacSCVε	MacSCVe	$\frac{\partial(v^2)}{\partial x} - v\frac{\partial(v)}{\partial x}$
		MacQCVε	MacQCVe	$\frac{1}{2}\frac{\partial(v^2)}{\partial x}$
		MacVDFε	MacVDFe	$\frac{\partial(\alpha\rho v^2)}{\partial x} - v\frac{\partial(\alpha\rho v)}{\partial x}$
	Conservative	MacCMε	MacCMe	
		MacALLOR		

Table 5.1: Notation to mention schemes depending on their momentum conservativity and their energy formulation. Conservativity : total energy - momentum - total energy and momentum

Note that on each output that will be presented, some additional curves will be shown :

- *Reference* a black bold plot, is the analytical solution (if it exists).
- VDF trust in sky blue, is the output from `TRUST` VDF scheme, which is the scheme we want to get as close as possible to. It is a semi-conservative-like scheme.
- PolyMAC, which also comes from `TRUST` and which is similar to VDF but for polyhedral meshes.

As it is a convention proceeding that way, only the density profile will be shown when trying to compare schemes between each other. Other profiles will be shown if necessary. Sometimes, a zoom is performed on plots, it is always centered on the very first contact discontinuity.

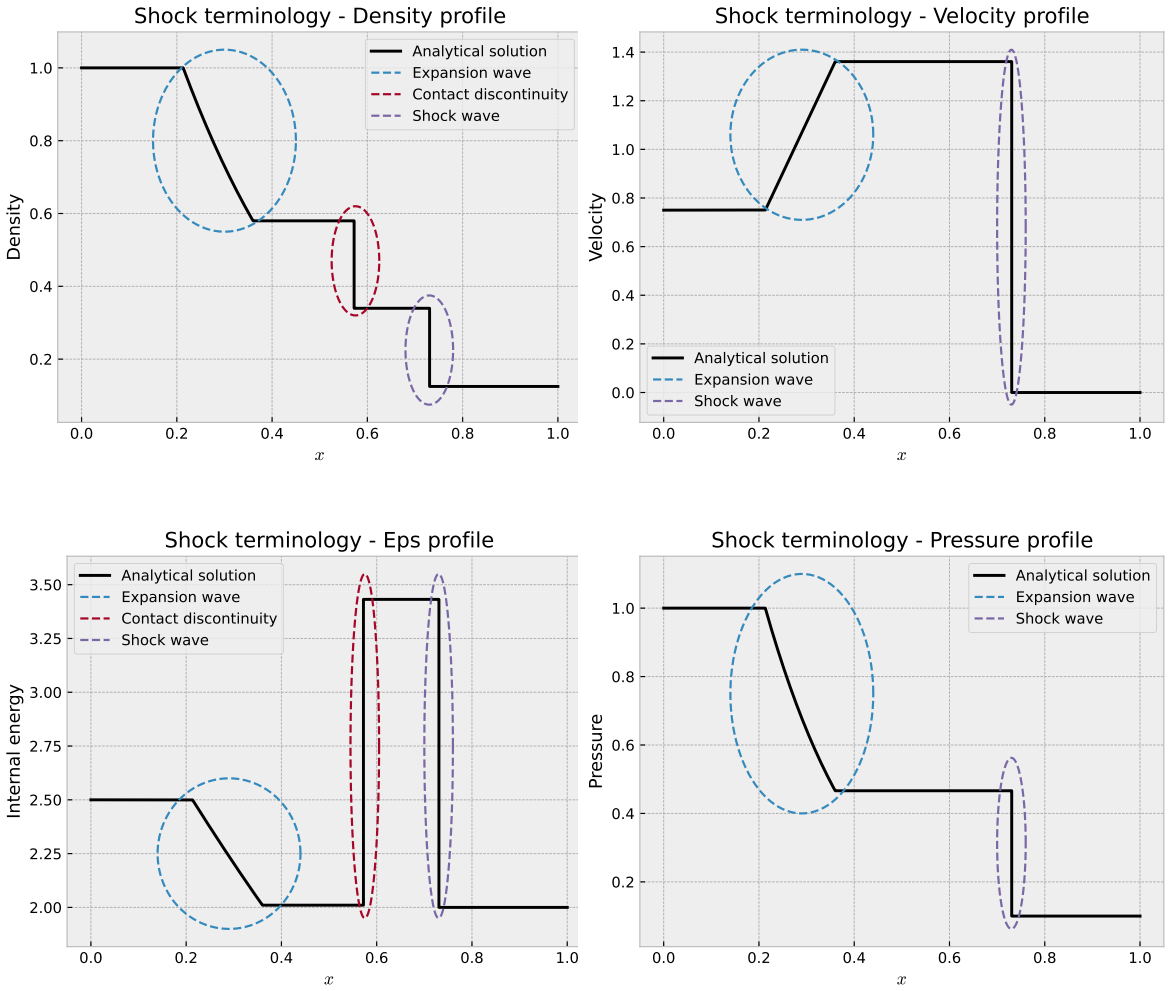
Remark In order to have a trustworthy comparison with `TRUST` numerical results, we force the time step to be constant with $\Delta t = 10^{-6}$ and cell number to be $N = 800$. This will enable to avoid having different parameters other than the schemes that can change.

Remark All the plots that are shown below have been produced with the scripts located in directory `hydromockup/studies/Report_post_processing_scripts`. I believe the best basis to create post-processing tools on your own is the script `script_1_preliminary_Mac_and_MacCM.py`.

5.1.2 Terminology

It is important to know what the used vocabulary designs. For this reason, plots of the density, velocity, internal energy and pressure for a shock tube problem are shown below. Ellipses show the regions we will call *expansion wave, contact discontinuity* and *shock wave* in the development.

Figure 5.1: Terminology for shocks on the example of an analytical solution of a shock tube test case



One thing that is major to note is that a contact discontinuity is a region where only density and internal energy are discontinuous. In fact it is the region where the two sides of the Riemann problem collide, thus have same pressure and speed, but obviously different densities.

5.2 Test cases

For notations, Ω is the domain, N the number of cells, γ the adiabatic coefficient and p_∞ the stiffened gas parameter.

5.2.1 Toro1 test case

Description It is a mono-fluid test case similar to a Sod shock tube, with a side of the Riemann problem with non zero velocity. It models perfect gas equation of state and does not require small CFL condition. Final time is chosen to be $t_f = 0.2$.

Motivation This test case is presented in E. Toro's book[3] and as it is a light shock (pressure ratio of 10^1) that `TRUST` also implements, it seemed a good starting point to check the code is working. Moreover, as the aim of this internship is to perform shock capture, it is natural to work on a shock test case. Finally as this test is quite common literature, it makes a good comparison point with other methods.

Ω	$[0, 1]$
N	800
CFL	0.2
γ	1.4
p_∞	0

Figure 5.2: Simulation parameters

	$0 \leq x \leq 0.3$	$0.3 \leq x \leq 1$
Density	1	0.125
Velocity	0.75	0
Int. energy	2.5	2

Figure 5.3: Initial conditions for Toro1 test case

Parameter p_∞ being null, the Stiffened Gas equation of state will be the very same as the perfect gas one. This choice is made to have a simple test case, easier to interpret. The adiabatic index value is chosen to be 1.4.[1]

5.2.2 Leblanc test case

Description This test case is a mono-fluid one, similar to Toro1. Its main difference resides in the strength of the shock it models as a logarithmic scale is required to plot the density profile, with a pressure ratio of 10^9. Final time is chosen to be $t_f = 6$.

Motivation As the shock is way stronger than Toro1's one, this case is generally hard to pass for most schemes and codes, meaning it can be a great way to push our schemes to their limits in term of shocks. Moreover, as the elapsed time is much longer, it also tests the stability of the scheme.

Ω	$[0, 10]$
N	800
CFL	0.2
γ	1.667
p_∞	0

Figure 5.4: Simulation parameters

	$0 \leq x \leq 3$	$3 \leq x \leq 10$
Density	1	10^{-3}
Velocity	0.1	0
Int. energy	0.1	10^{-7}

Figure 5.5: Initial conditions for Toro1 test case

Again this test case uses the perfect gas equation of state since p_∞ is null. The adiabatic coefficient is set at 1.667.[2]

[1] Value $\gamma = 1.4$ could refer to a large variety of species : dry air, hydrogen, nitrogen, oxygen, carbon monoxide, fluorine or chlorine, at room temperature, 20°c.

[2] Value $\gamma = \frac{5}{3}$ could refer mostly to perfect gas in the common meaning : helium, neon, argon, krypton, xenon or radon at room temperature, 20°c.

5.2.3 Gaussian advection test case

Ω	$[-5, 25]$
N	800
CFL	0.2
γ	1.4
p_∞	0

Figure 5.6: Simulation parameters

Description This is a density profile transport problem, which is initialized as a gaussian function. Numerical error is reached before touching domain's border. Pressure and velocity are uniform over the domain and classic Stiffened Gas EOS is used. Final time is $t_f = 10$. Periodic boundary condition is imposed.

Motivation This test case is made in order to have a smooth problem with a known analytical solution (for density profile). The aim is to be able to perform order checks on the schemes as it contains no shocks.

Initial profiles This test case is all about the density profile, but all variables need to be initialized, which is done as follows :

$$\rho_0(x) = \frac{1}{2} + \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5.1}$$

$$(\rho\varepsilon)_0(x) = \frac{1}{\gamma - 1} p_0(x) \tag{5.2}$$

$$p_0(x) = 1 \tag{5.3}$$

$$v_0(x) = 1 \tag{5.4}$$

where $(\mu, \sigma) = (5, 1.0)$, chosen so that computer error is reached by the gaussian function before touching the borders and so that the gaussian profile is not too steep.

Note that for an arbitrary time t, the analytical solution of the density profile advection problem is known and can be expressed thanks to the initial profile as

$$\rho(t, x) = \rho_0(x - v_0 t) \tag{5.5}$$

i.e. the initial gaussian profile, transported during some time t at velocity v_0. Then, final time t_f can be chosen using the relation

$$t_f = \frac{x_f - x_0}{v_0} \tag{5.6}$$

where $x_0 = \mu$ in our previous notations, and x_f is the x-coordinate on which one wants to have the transported gaussian centered. For instance, with $x_0 = 5$, $x_f = 15$ and $v_0 = 1$, we get $t_f = 10$.

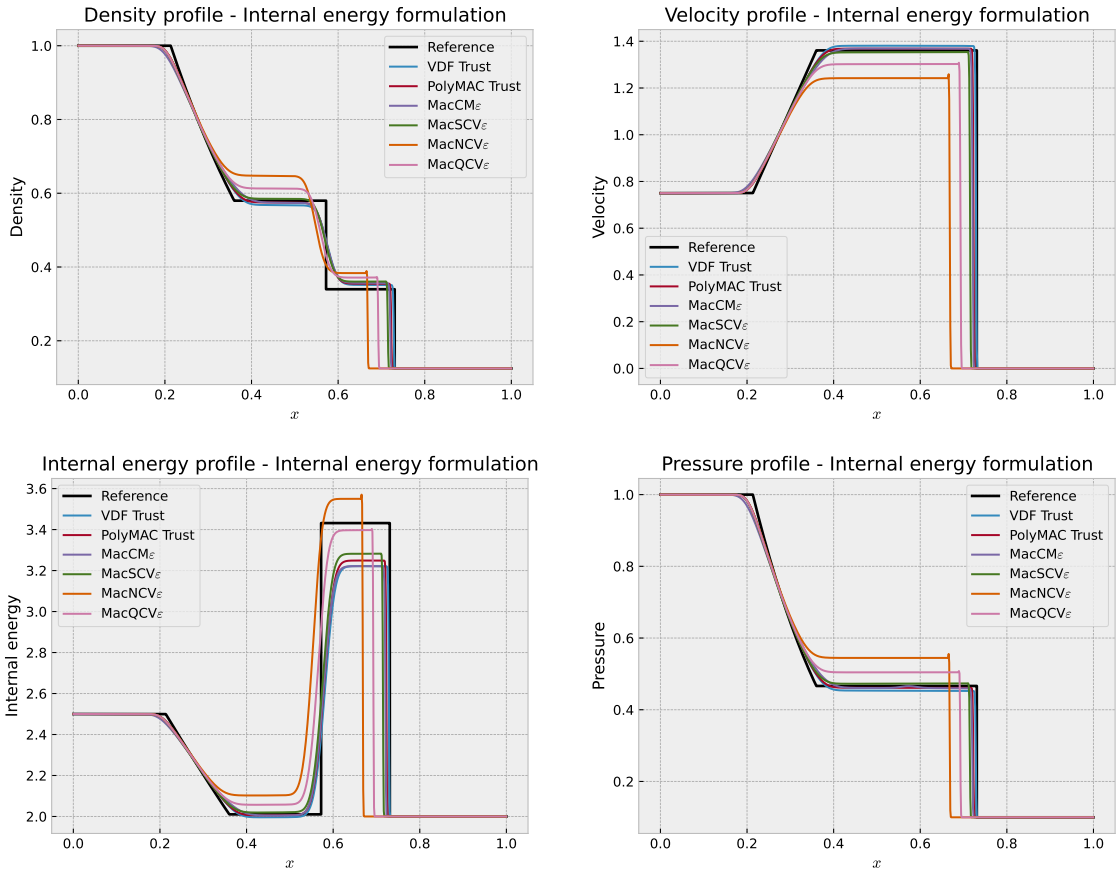
5.3 Preliminary results

In this section, we focus on presenting simple and straight forward results to get an intuition on each scheme behaviour. For this section, **all interpolations are done upwind except all needed density terms in momentum equation which are taken fully centered**.

5.3.1 Overview of the schemes

Internal energy formulation Working on test case Toro1, using schemes MacNCVε, MacSCVε, MacQCVε and MacCMε, the code generates the plots below. The volumic fraction is not shown as it is a mono-fluid test case, but it is indeed constant at 1 over the domain. The plots have been generated with script `script_1_preliminary_Mac_and_MacCM.py`.

Figure 5.7: Toro1 test case under internal energy formulation - plot of the variables of the problem at $t = 0.2$ for basic schemes

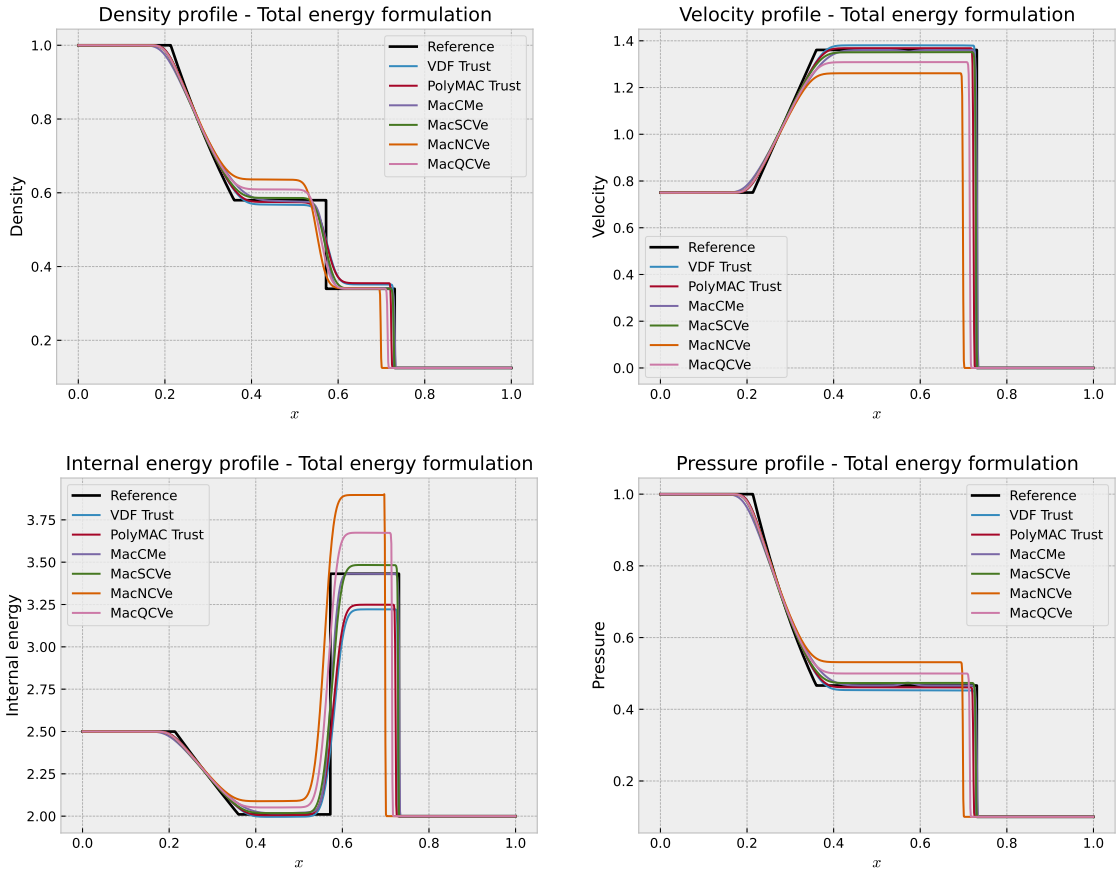


Schemes MacQCVε and MacNCVε have really bad performances as their plateaus and drops are not in-sync at all ; we will abandon them from here. It shows the importance of how the term $v\frac{\partial(v)}{\partial x}$ is discretized, even with equal continuous formulations.

Even if MacCMε and MacSCVε are performing better, a slight shift can be observed on the shock wave as well as some heavy diffusion on the contact discontinuity. However, the contact discontinuity plateau is very well captured by MacSCVε, even better than by `TRUST` schemes. Some point thus have to be improved to increase our scheme performances or to get closer to `TRUST` outputs.

Total energy formulation In this formulation, the specific internal energy equation is replaced by a specific total energy equation which is supposed to be conservative. This might impact the overall shock capture. Note that test case Toro1 is still used for this illustration, and script `script_1_preliminary_Mac_and_MacCM.py` can be used to reproduce these plots.

Figure 5.8: Toro1 test case under total energy formulation - plot of the variables of the problem at $t = 0.2$ for basic schemes



The major difference is that all schemes appear to have a great phase with the shock wave as well as a great estimation on the related plateau's level. Some diffusion can still be observed on the contact discontinuity, and no improvement is to be seen on the expansion wave.

On the other hand, it is possible to see that MacQCVe and MacNCVe are still performing badly, meaning we will let them go from now on. Moreover, note that MacCMe which is the fully conservative scheme has a very good accuracy but seems to have less precision on the smooth part of the problem, the expansion wave.

Overall, all schemes seems to be performing better under total energy formulation. We won't take this as a standard since `TRUST` does not use total energy formulations, but it can be kept in mind to improve results. This improvement is easily explained by the conservativity of the schemes under this formulation.

Remark Some zooms on the first contact shock are available at appendix I, for total energy formulation.

5.3.2 Conservativity checks for total energy formulation

In this development, we try to check whether our discretization and implementation preserve the density, momentum and total energy, as we know that Euler's equations do. We will use Toro1 test case to test conservativity.

Notations

For this purpose, we are going to compute the total value of these quantities over the spatial domain, and see how they evolve in time. As we wish to study the same property for some quantity $\Phi \in \{\rho, \rho v, \rho e\}$, we define the positive relative error term by :

$$|E_\Phi(t)| = \left| \frac{\int_\Omega (\Phi - \Phi_{th})(t,x)dx}{\int_\Omega \Phi_{th}(t,x)dx} \right|$$

The term that has no subscript will be the one we compute thanks to our code and the one with the subscript th will be a theoretic one. For a detailed explanation of the computation of the theoretic terms, see appendix J.

Outputs

Using the methodology that has been presented just before, we produce some plots using an altered `Python` script : `script_2_conservativity_plots.py`. We use the total energy formulation in order to check total specific energy conservation, but the very same observations can be made with the internal energy formulation, total energy excepted.

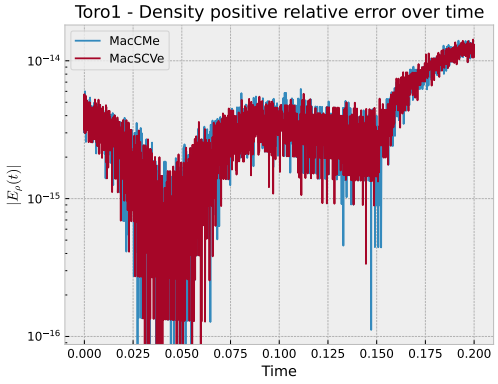


Figure 5.9: Density positive relative error

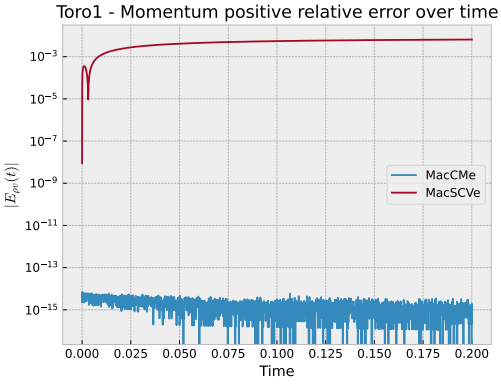


Figure 5.10: Momentum positive relative error

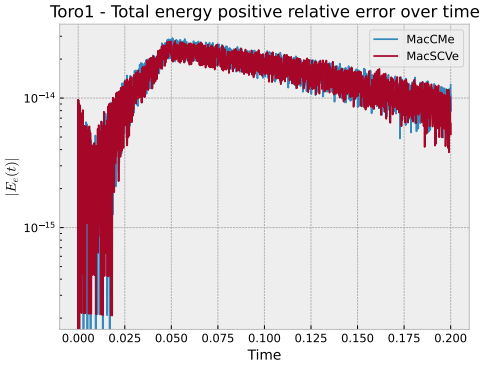


Figure 5.11: Total energy positive relative error

We do obtain what we expected : MacCMe is fully conservative but MacSCVe does not preserve momentum. Conservativity results for internal energy formulation can be found at appendix K.

Numerical interpretation The order of the values can be surprising as it is mainly around 10^{-15}. In fact, according to `Python` documentation[22], floating point numbers in `Python` use double precision, composed of a 52 bits mantissa[3]. This means the smallest positive real number that can be represented is $2^{-52} \approx 10^{-15.65}$; these conservativity plots reach numerical error. Knowing this bias is important in case one wants to extend the `C++` numerical core to deal with a greater precision as a `Python` post processing with default double precision will no be sufficient. A custom floating point number will then have to be used in the `Python` script. On the shape of the curves, we believe no additional mathematical interpretation can be made as numerical precision is reached.

5.3.3 Playing with artificial viscosity

This is for those like me at first, who do not know how artificial viscosity acts on the output result. Using internal energy formulation, internal energy profiles are shown as the effects are the most visible on these ones. Similar behaviours can be witnessed under total energy formulation. Those plots have be realized with the script `script_3_playing_with_viscosity.py`.

Figure 5.12: Internal energy changes depending on quadratic artificial viscosity value for MacSCVε

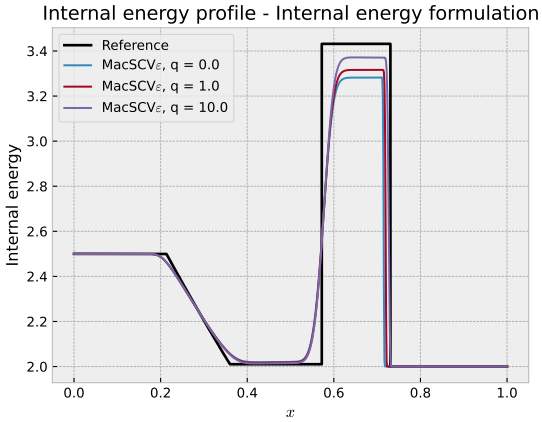
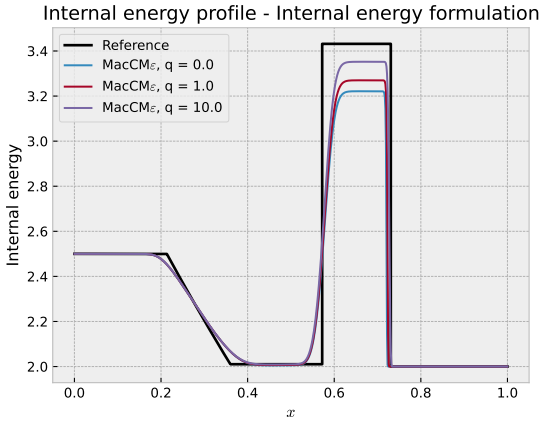


Figure 5.13: Internal energy changes depending on quadratic artificial viscosity value for MacCMε



It appears quadratic artificial viscosity helps the scheme to converge. As it increases, the approximation gets closer to the analytical solution. In fact it acts on the scheme stabilization in case of shocks, particularly for strong shocks. We do not plan on using linear viscosity as it acts on oscillations, which is an issue we do not have thanks to the upwind interpolation as it already introduces linear viscosity terms.

[3]To be exact, double precision uses 1 bit for sign, 11 for exponent and 52 for mantissa.

5.4 Advanced results

With some discussion with TRUST members and some consideration, we decided to try altering interpolations and discretizations in our schemes, which are presented hereinafter. Keep in mind that some bench-marking might have to be performed but we will focus on Toro1 test case.

5.4.1 Mixed momentum density interpolation for MacSCV

This is one of the schemes we worked the most on as it is the closest to TRUST ones. After a meeting with Antoine Gernschenfeld, instead of interpolating density fully centered, we apply this method

$$\begin{aligned}
&\Delta x_{i+\frac{1}{2}} \left[\alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}}\right]_c \frac{v^{k,n+1}_{i+\frac{1}{2}} - v^{k,n}_{i+\frac{1}{2}}}{\Delta t} \\
&+ \left[\alpha^k_{i+\frac{1}{2}} \rho^k_{i+\frac{1}{2}}\right]_u \left(\left[v^k_{i+1}\right]_u{}^2 - \left[v^k_i\right]_u{}^2 - v^k_{i+\frac{1}{2}} \left(\left[v^k_{i+1}\right]_u - \left[v^k_i\right]_u \right) \right) \\
&+ \left[\alpha^k_{i+\frac{1}{2}}\right]_u \left(p^k_{i+1} - p^k_i\right) = 0
\end{aligned} \tag{5.7}$$

where advection-linked density is chosen upwind and time-linked is chosen centered. This is how the American code TRACE[18] does, which TRUST is inspired from. This methods actually accounts for the different natures of the term (transport vs time integration). By the way, this is the very first definition we provided for MacSCV in the numerical development at the beginning of this report.

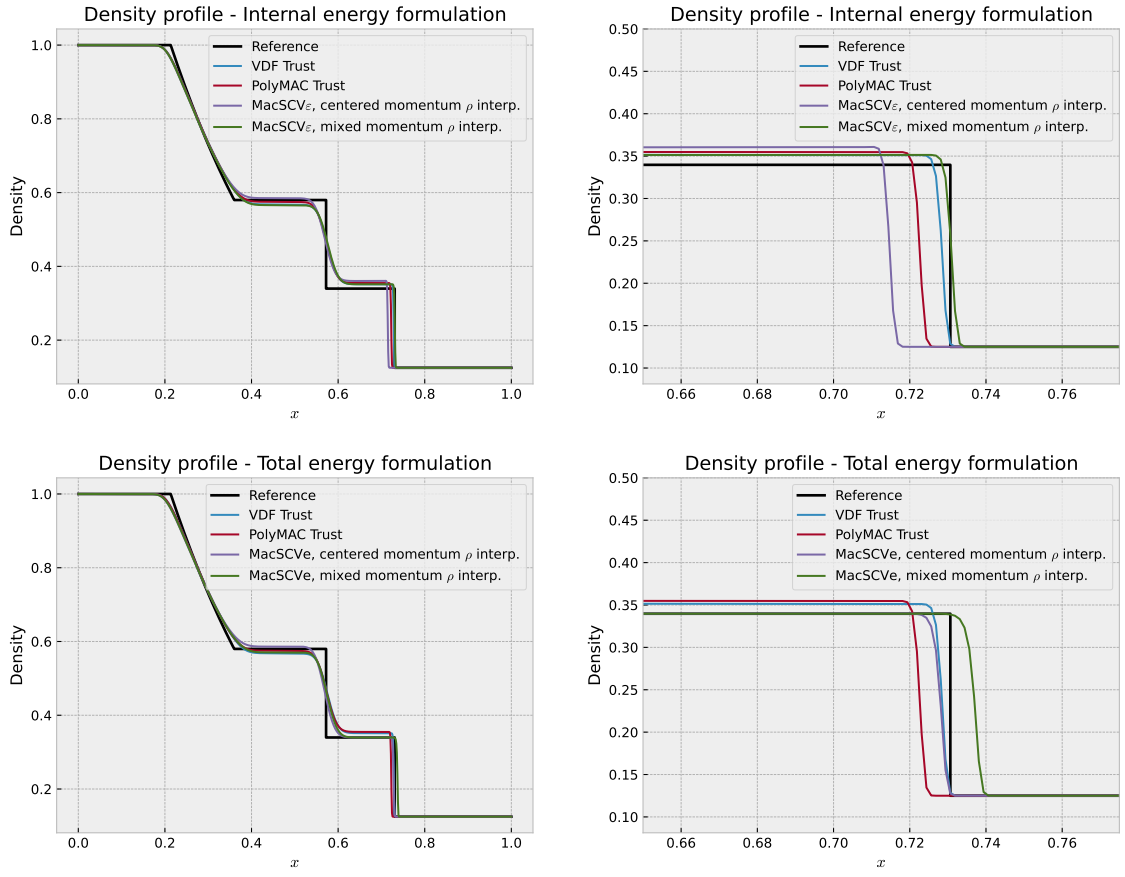


Figure 5.14: Toro1 test case. On the left, classic plots of density profiles, on the right plots, zoom on the first contact shock. First line is under internal energy formulation, second line is under total energy formulation.

Note that these plots have been made with `script_4_MacSCV_split_rho.py`. This method improves the approximation in internal energy formulation case whereas it seems to degrade the result in case of total energy formulation. This is likely to be due to the introduction of dispersion error terms with the upwind density terms that account for some of the missing kinetic energy. This means that when working in the ε version of the schemes, it adds something missing so results are better whereas when working with e versions, it adds something that the total energy equation already accounts for i.e. it is an additional error that is not helping. From now on, we use MacSCVε with mixed momentum density interpolation. It fulfills the goal of the internship to reproduce or improve `TRUST` outputs.

5.4.2 Kinetic energy interpolation for MacSCVe

Beware, the scheme is considered under its total energy formulation in this paragraph. We tried to alter other terms than momentum transport, such as the internal energy. In fact, working under e version of the scheme, internal energy has to be derived from total and kinetic energies. We thought about two ways to proceed :

$$\varepsilon_i^k = e_i^k - \frac{1}{2}\left(\frac{v_{i+\frac{1}{2}}^k + v_{i-\frac{1}{2}}^k}{2}\right)^2 \quad \text{or} \quad \varepsilon_i^k = e_i^k - \frac{1}{2}\left(\frac{(v_{i+\frac{1}{2}}^k)^2 + (v_{i-\frac{1}{2}}^k)^2}{2}\right) \tag{5.8}$$

What we observe is that there is no difference between those two terms when using constant time step or not. However the use of automatic time step induce a slight shift of the approximation of the first contact shock to the right. See `script_5_MacSCV_totenergy_kinetic_tests.py`.

5.4.3 Pressure gradient interpolation for MacSCVe

The pressure gradient in the total energy equation has to be interpolated, we recall here the total energy equation discretization, with pressure gradient interpolation undefined :

$$\begin{aligned}
&\Delta x_i \frac{\alpha_i^{k,n+1}\rho_i^{k,n+1}e_i^{k,n+1} - \alpha_i^{k,n}\rho_i^{k,n}e_i^{k,n}}{\Delta t} \\
&+ \left[\alpha_{i+\frac{1}{2}}^k \rho_{i+\frac{1}{2}}^k e_{i+\frac{1}{2}}^k\right]_u v_{i+\frac{1}{2}}^k - \left[\alpha_{i-\frac{1}{2}}^k \rho_{i-\frac{1}{2}}^k e_{i-\frac{1}{2}}^k\right]_u v_{i-\frac{1}{2}}^k \\
&+ \left[\alpha_{i+\frac{1}{2}}^k\right]_u \left[p_{i+\frac{1}{2}}^k\right]_* v_{i+\frac{1}{2}}^k - \left[\alpha_{i-\frac{1}{2}}^k\right]_u \left[p_{i-\frac{1}{2}}^k\right]_* v_{i-\frac{1}{2}}^k + \Delta x_i p_i \frac{\alpha_i^{k,n+1} - \alpha_i^{k,n}}{\Delta t} = 0
\end{aligned} \tag{5.9}$$

As pressure is not a conserved quantity, it has different nature as density or energy.

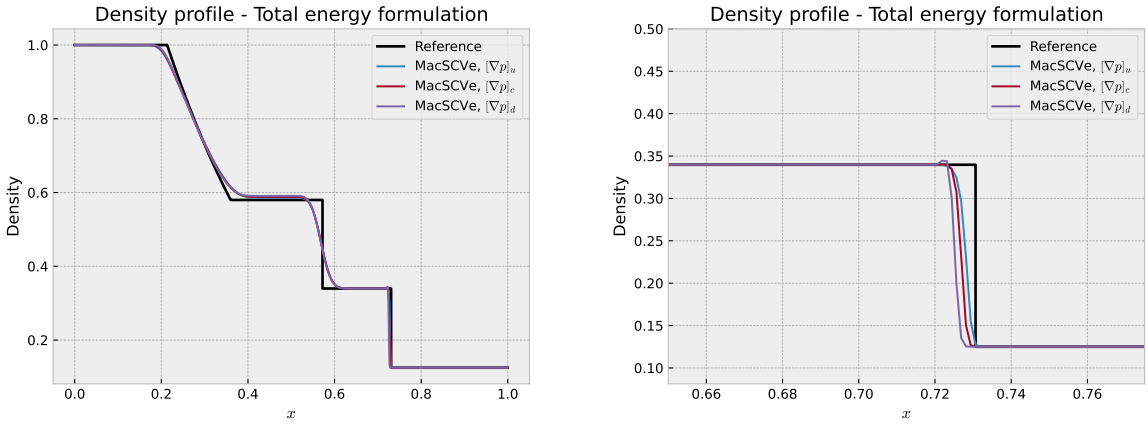


Figure 5.15: Density profile for Toro1 test case approximated with MacSCVe.

It is possible to see that there is not big effect on the outcome. However, the plateau liked to the expansion wave seems to be approximated the best for upwind pressure gradient, which is consistent physically. On the other hand, the shock wave appears to be great with upwind method.

This means that upwind method is indeed the best, for this test case at least. When dealing with MacSCVe, pressure gradient will then be kept upwind in the following development. Note that the script that has been used to produce these plots is `script_6_Pressure_gradient_for_MacSCV.py`

5.4.4 Momentum transport interpolation in MacCM

Recall that the momentum equation for MacCM scheme is discretized as

$$\begin{aligned} &\Delta x_{i+\frac{1}{2}} \frac{\left[\alpha^{k,n+1}_{i+\frac{1}{2}} \rho^{k,n+1}_{i+\frac{1}{2}}\right]_c v^{k,n+1}_{i+\frac{1}{2}} - \left[\alpha^{k,n}_{i+\frac{1}{2}} \rho^{k,n}_{i+\frac{1}{2}}\right]_c v^{k,n}_{i+\frac{1}{2}}}{\Delta t} \\ &+ \alpha^k_{i+1}\rho^k_{i+1}\left[v^k_{i+1}\right]_*\left[v^k_{i+1}\right]_* - \alpha^k_i\rho^k_i\left[v^k_i\right]_*\left[v^k_i\right]_* \\ &+ \left[\alpha^k_{i+\frac{1}{2}}\right]_u (p_{i+1} - p_i) = 0 \end{aligned} \tag{5.10}$$

where interpolation for $\left[v^k_i\right]_*\left[v^k_i\right]_*$ has to be properly defined. In an internal paper[4] about PolyMAC (Mark-and-Cells scheme implemented in `TRUST` for unstructured meshes), A. Gerschenfeld uses a custom method to interpolate transport terms. He cuts the transported quantity and the velocity of transport in two parts, respectively interpolating the first with upwind method and the second with centered fashion. See `script_7_MacCM_totenergy_velocity_tests.py` and the `C++` source code about MacCM to give it a try.

Internal energy formulation On the left, the velocity is interpolated as $\left[v^k_i\right]_u\left[v^k_i\right]_u$ and on the right as $\left[v^k_i\right]_u\left[v^k_i\right]_c$. Mixed interpolation seems to introduce some oscillations, which is due to the centered interpolation, but the shock approximation has a better phase when looking at the analytical solution.

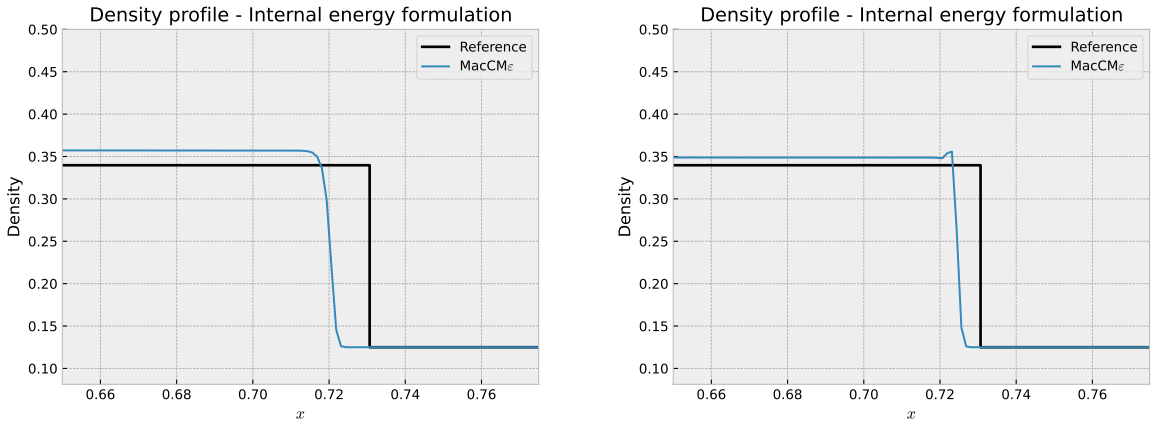


Figure 5.16: Toro1 with MacCMε. On the left, fully upwind velocities, on the right, upwind conserved quantities and centered transport velocity. These are zooms on the shock wave.

Total energy formulation Again, on the left is the fully upwind interpolation and on the right is the mixed one. Under total energy formulation, oscillations do not appear when using mixed

[4] A. Gerschenfeld, Y. Gorsse, 2021, *Development of a robust multiphase flow solver on general meshes ; Application to sodium boiling at the subchannel scale*, Université Paris-Saclay, CEA, Service de Thermo-hydrolique et de Mécanique des Fluides

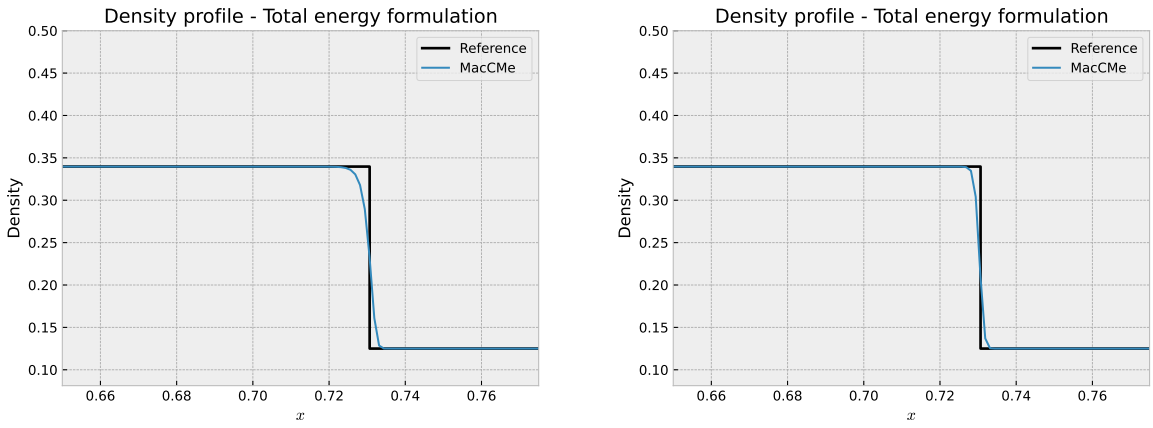


Figure 5.17: Toro1 with MacCMe. On the left, fully upwind velocities, on the right, upwind conserved quantities and centered transport velocity.

interpolation. This is likely to be due to energy conservation, thus probably to kinetic energy or internal energy conversion. Moreover, total energy case shows a sharper capture of the shock, which is a behaviour that could be very useful to increase precision.

5.4.5 Kinetic energy interpolation for MacCMe

As we have done for MacSCVe, it is possible to choose different ways of getting internal energy back

$$\varepsilon_i^k = e_i^k - \frac{1}{2}\left(\frac{v_{i+\frac{1}{2}}^k + v_{i-\frac{1}{2}}^k}{2}\right)^2 \quad \text{or} \quad \varepsilon_i^k = e_i^k - \frac{1}{2}\left(\frac{(v_{i+\frac{1}{2}}^k)^2 + (v_{i-\frac{1}{2}}^k)^2}{2}\right) \tag{5.11}$$

These two methods show no difference on the output result, both with constant time step or adaptive one.

5.4.6 Pressure gradient interpolation for MacCMe

In the same idea as for MacSCVe, it is possible to try interpolating the pressure gradient in different ways. Plots have been generated with `script_9_Pressure_gradient_for_MacCM.py`.

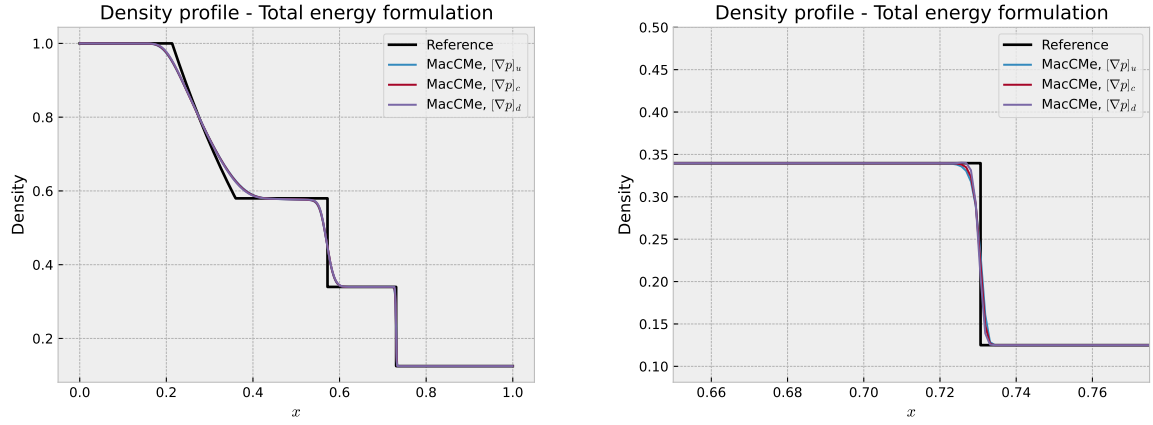


Figure 5.18: Toro1 with MacCMe. Density profile, zoomed on the shock wave (right), comparison of the different gradient interpolations (upwind, centered, downwind).

We observe almost no difference between the different choices. There is a small improvement on the shock wave, where the use of a downwind pressure gradient sharpens slightly the shock and is less diffusive.

5.5 Miscellaneous results

Here are presented some results that could not fit in the previous parts but that remain very interesting both for their performance or their mathematical aspects.

5.5.1 MacVDFε performance

Recall that this scheme is different from MacSCVε because of its momentum equation continuous formulation. In fact, it keeps momentum as a whole variable inside transport terms, as it is done in `TRACE`[18]. This might be the closest formulation of the scheme to what is done in `TRUST` but it is only a guess as no documentation of it exists.

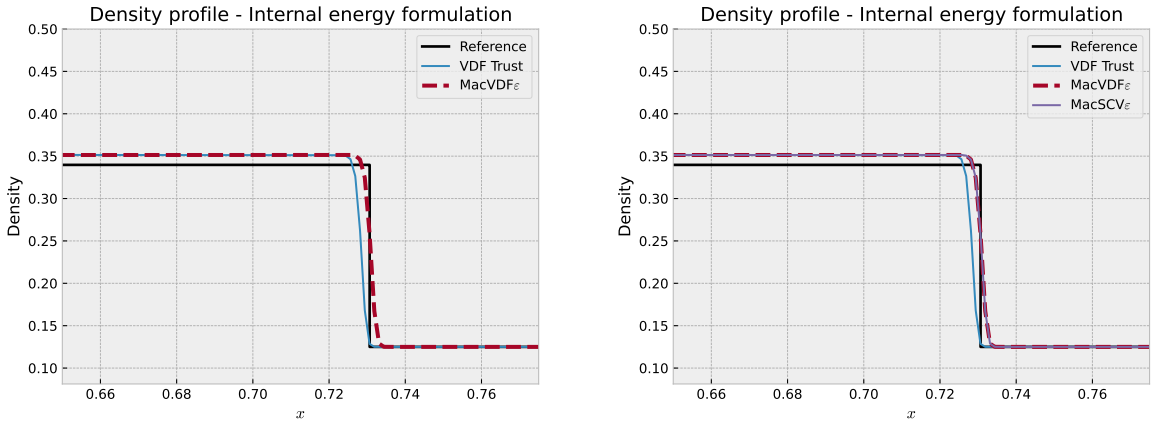


Figure 5.19: Toro1 with MacVDFε. Density profile, zoomed on the shock wave. On the left comparison with MacSCVε with mixed density interpolation in momentum

We witness on the left plot that this scheme gets really close to the `TRUST` results for VDF `TRUST` ; a zoom is performed on the shock wave as it is a critical region, but the schemes have almost exact same results.

On the right plot are compared MacSCVε and MacVDFε, which have strictly different discretizations. It is possible to see that they seem to produce equal approximations. With these two plots, the goal of the internship is reached as we managed to reproduce `TRUST` output and even to improve it. This is at the cost of having a fully explicit scheme for now, with fixed time step $\Delta t = 10^{-6}$. Plots have been produced with `script_10_MacVDF.py`.

5.5.2 MacALlor results

This is the scheme that has been provided by Antoine Llor, which can be found in appendix E and its total energy conservativity proof in appendix F.

It seems useful to recall that this scheme only uses upwind interpolation for all variables. Thanks to the diffusion term D_i^n, this scheme is conservative for total energy and should capture plateaus very well. See script `script_11_MacALlor.py` to reproduce those results.

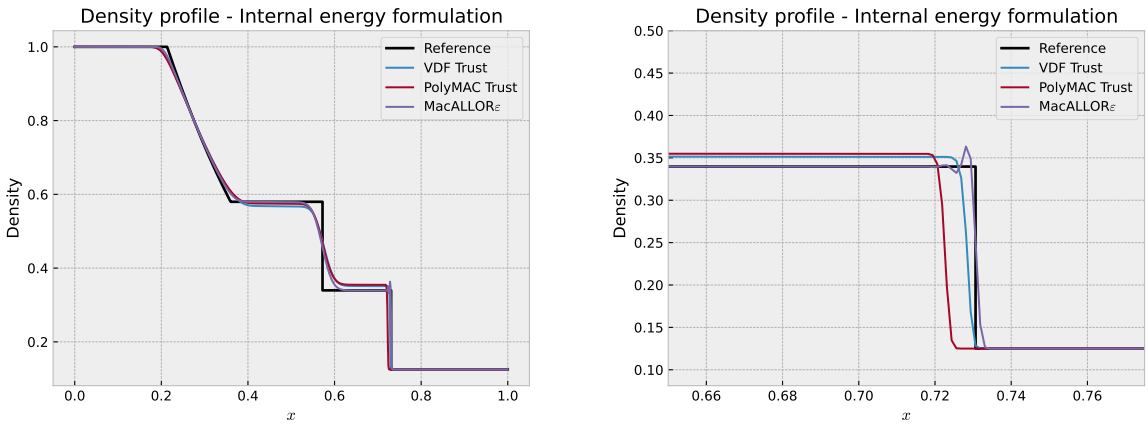


Figure 5.20: Toro1 with MacALlor. Density profile, zoomed on the shock wave on right plot. Comparison with PolyMAC and VDF from the CEA code.

We see that indeed this scheme is very efficient as plateaus and Hugoniot relations are verified. However, this is at the cost of creating oscillations on the shock wave, which can be witnessed on all other variables plots. Thanks to the use of artificial viscosity, these oscillations can be taken down.

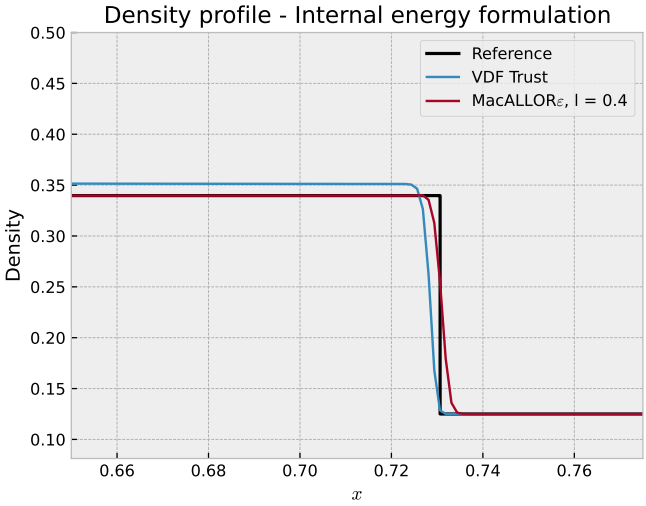


Figure 5.21: Density profile on Toro1 test case with MacALlor scheme with linear viscosity 0.4. Zoom on the shock wave.

As this is a slight oscillation but the shock is stable, we use only linear viscosity. The effect is not deniable. This scheme is very efficient as even without artificial viscosity, it is more efficient than our previous ones such as MacSCV or MacCM, both in their internal and total energy formulations.

The frustrating part is that this scheme would require a heavy work to be implemented in the `TRUST` plateform as it has a very different nature. For now, we only work under 1D framework, but `TRUST` can go up to 3D, meaning mathematical and programming issues such as numerical error management, stability and validation begin to be quite complicated.

5.5.3 Leblanc test case with best MacSCV version for Toro1

Leblanc test case has been introduced at the beginning of this chapter and is a strong shock. To test it we will work with my custom script `script_12_Leblanc_test_case.py`. We use the version of MacSCV that has mixed interpolation for its density terms in the momentum equation.

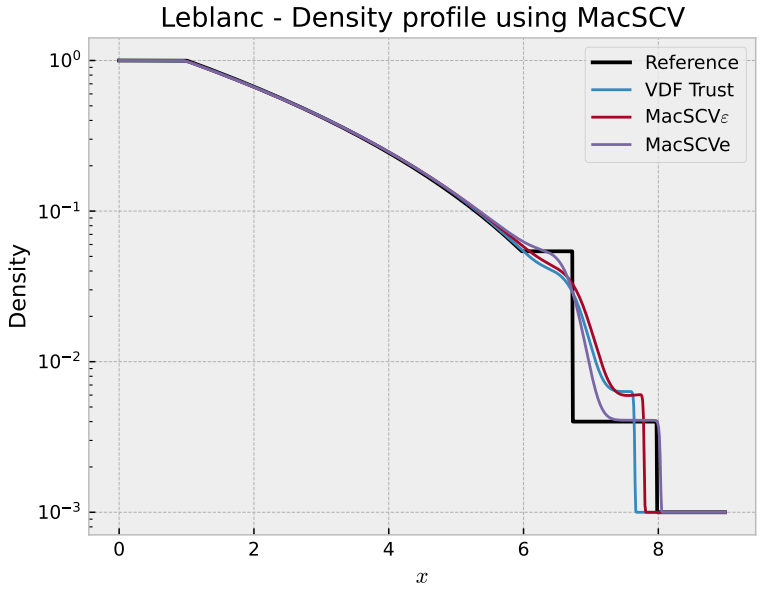


Figure 5.22: Density profile with leblanc test case using MacSCV scheme under internal and total energy formulations.

Here are plotted the results for the specific internal and total energy formulations of the scheme. The internal formulation, which we found to be similar to VDF from `TRUST` earlier, seems to still have slightly better performances as the shock wave approximation appears closer. On the total energy formulation hand, even if the contact discontinuity is not well captured, the shock wave is.

In order to have a comparison point with `TRUST` we chose a fixed time step $\Delta t = 10^{-4}$, bigger than before (10^{-6}) since this test case runs on a longer period. Moreover, our current adaptive time step blows up on this test case[5].

5.5.4 Second order accuracy extension

It remains to try our second order methods. In fact, even in the very first Mac scheme, the pressure gradient is of order 2, meaning it is likely that what we considered to be a method of order 1 is actually intermediary between order 1 and 2.

To get the order of the scheme, we use a smooth test case, the Gaussian advection one. It is useful as the exact solution is known so the approximation error can be known exactly. This test case is repeated for different numbers of cells : $N \in \{2^k, k \in [\![5; 15]\!]\}$ i.e. $N = 32, \dots, 32768$. We use the L^∞ norm[6] to compute the error, meaning we only focus on the very top of the gaussian curve.

We see that the order 1 method seems to have a true order 1 for convergence, meaning it passes this test. However order 2 is a bit deceiving as this test case is smooth meaning that limiters should

[5]First computed times steps are of order $10^{-2}, 10^{-3}$ but it then drops to less than 10^{-12} which is the threshold value we have chosen to make the code exit.

[6]Recall that for a vector $v \in \mathbb{C}^n$ with $v = (v_1, \dots, v_n)$, this norm is defined as $\|v\|_\infty = \max_{1 \leq i \leq n} |v_i|$.

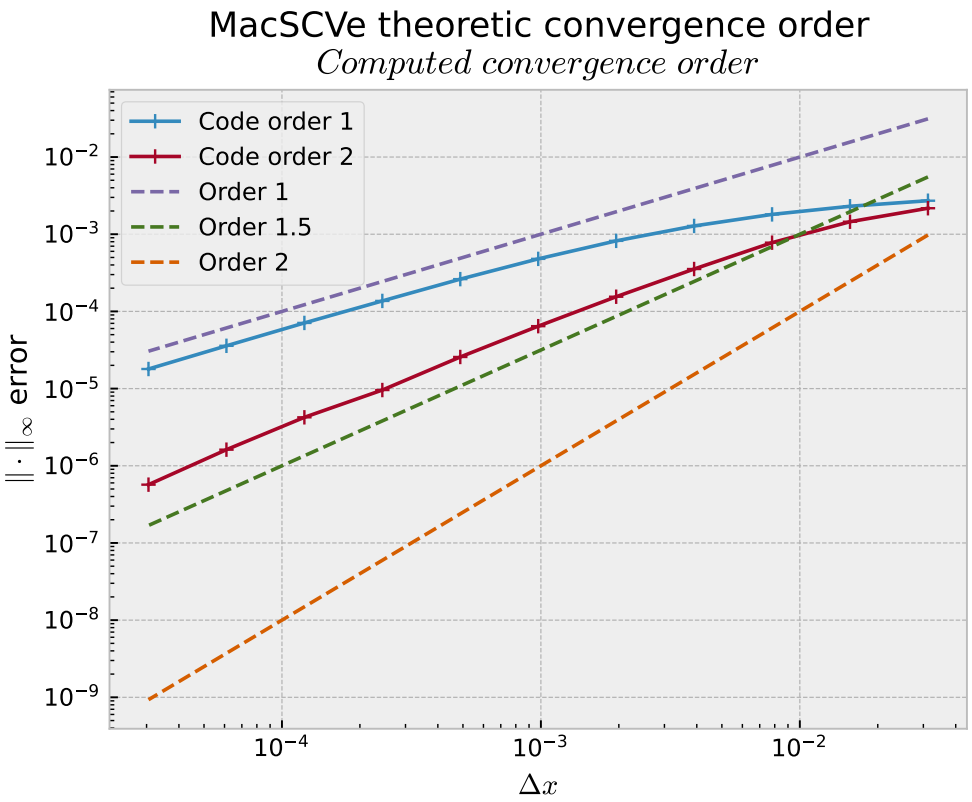


Figure 5.23: MacSCV*e* convergence order plots.

not activate. In fact, having an order of 1.5 instead of 2 for a smooth case means that limiters are activating, but they should not.

These results are not satisfying, but time is missing in my internship to further investigate this issue. Note that to reproduce this order plots, script `script_13_Gaussian_advection_order2.py` should be used or modified.

Chapter 6

Conclusion and outlooks

6.1 Outlooks

As this report is very long, here is a list of possible tracks to follow for pursuing the code development as well as the schemes investigations :

- Validate the code on a wide range of test cases (sedimentation, double relaxation, complex transport problems, stronger shocks, gravity etc.)
- Check for missing gravity source terms as we developed the homogeneous version of the equations to simplify the study.
- Implement closure conditions to extend the code to multi-fluid problems. The code has been developed with this aim, but closures and test cases are missing for now.
- Solve order 2 issues.
- Extend the schemes to 2D or 3D frameworks.
- Extend the schemes to unstructured meshes, or if in 1D, to irregular ones.
- Use implicit or ICE time incrementation.

6.2 Conclusion

This internship enabled to develop a large variety of numerical schemes to solve mono-fluid Euler's equations under internal or total energy formulation. The difference between those schemes is either how the equations are written and modified under their continuous form, or the way variables are interpolated or reconstructed.

For instance, the main goal of the internship that was to approach `TRUST` results has been reached with the use of a so-called semi-conservative scheme, MacSCVε (and MacVDFε). This scheme even goes beyond VDF `TRUST` performances as it is centered on the shock wave in the case of Toro1 test case.

This internship also showed that taking care of conservativity is very important when dealing with shocks. In fact, all schemes seems to produce better results under their most conservative versions.

Having in mind this internship is driven by the constraint of changing `TRUST` schemes in simple ways, it seems difficult to really improve results while keeping non conservative formulations. For instance, even if Godunov schemes are designed to work with shocks, Mac schemes were designed for non compressible modes and do perform better in this case.

Appendix A

Conservativity equivalency proof

Let $\Omega \subset \mathbb{R}^n$ closed and $\mathcal{T} = [0, T]$ a time interval with $T > 0$. Let $\phi : \mathcal{T} \times \Omega \to \mathbb{R}^n$ be a mesurable function with respect to $x \in \Omega$, differentiable with respect to $t \in \mathcal{T}$, and assume there exists a measurable function $\psi : \Omega \to \mathbb{R}^n$ such that $\forall t \in \mathcal{T}, \|\frac{\partial(\phi)}{\partial t}(t, x)\| \leq \|\psi(x)\|$ for any norm suiting $\mathbb{R}^n$. Such ϕ will represent the quantity that is to be conserved. Moreover, let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a $\mathcal{C}^1(\mathbb{R}^n)$ function representing the flux. Then composition $F \circ \phi$ is well defined.

Assume the following relation holds between ϕ and its flux through the boundary :

$$\frac{\partial}{\partial t} \int_\Omega \phi(t, x)dx = -\int_{\partial\Omega} F(\phi(t, x)) \cdot n dx \tag{A.1}$$

This relation is well defined as ϕ is measurable on $\mathbb{R}^n$. As ϕ is differentiable with respect to time and its time partial derivative is uniformly bounded by a measurable function of Ω, which we called ψ in our hypothesis, derivative and integral can be inverted. This process is applied on the left member of previous equality leading to :

$$\frac{\partial}{\partial t} \int_\Omega \phi(t, x)dx = \int_\Omega \frac{\partial(\phi)}{\partial t}(t, x)dx \tag{A.2}$$

On the other hand, as the composition $F \circ \phi$ is well defined, Ω being closed and $F \in \mathcal{C}^1(\mathbb{R}^n)$, the *Green-Ostrogradski* theorem can be used to turn the boundary integral into a whole Ω one, for the right member of the very first equation :

$$-\int_{\partial\Omega} F(\phi(t, x)) \cdot n dx = -\int_\Omega \nabla_x \cdot F(\phi(t, x))dx \tag{A.3}$$

It lasts to put both transformed terms from equation A.2 and A.3 together to make appear the weak formulation of the relation :

$$\int_\Omega \left(\frac{\partial(\phi)}{\partial t}(t, x) + \nabla_x \cdot F(\phi(t, x)) \right) dx = 0 \tag{A.4}$$

Finally, as this relation holds for any closed $\Omega \in \mathbb{R}^n$, the integral sign can be taken down to get :

$$\frac{\partial(\phi)}{\partial t}(t, x) + \nabla_x \cdot F(\phi(t, x)) = 0 \tag{A.5}$$

To conclude on the equivalence, the reciprocal follows directly from the smoothness and differentiability properties of the function, using the very same theorems, meaning that :

$$\frac{\partial(\phi)}{\partial t}(t, x) + \nabla_x \cdot F(\phi(t, x)) = 0 \iff \frac{\partial}{\partial t} \int_\Omega \phi(t, x)dx = -\int_{\partial\Omega} F(\phi(t, x)) \cdot n dx \tag{A.6}$$

Appendix B

Euler's total energy formulation

The overall idea is to use the Momentum to build a the kinetic energy equation. Then the addition of kinetic and Internal Energy equations builds the total energy equation as no other form of energy has to be considered here.

Start by multiplying the Momentum equation by v to get

$$v\frac{\partial(\rho v)}{\partial t} + v\frac{\partial(\rho v^2)}{\partial x} + v\frac{\partial(p)}{\partial x} = 0 \tag{B.1}$$

Notice that :

- $v\frac{\partial(\rho v)}{\partial t} = \frac{\partial(\rho v^2)}{\partial t} - \rho v\frac{\partial(v)}{\partial t}$
- $v\frac{\partial(\rho v^2)}{\partial x} = \frac{\partial(\rho v^3)}{\partial x} - \rho v^2\frac{\partial(v)}{\partial x}$

Replace corresponding terms in equation B.1 to get

$$\frac{\partial(\rho v^2)}{\partial t} - \rho v\frac{\partial(v)}{\partial t} + \frac{\partial(\rho v^3)}{\partial x} - \rho v^2\frac{\partial(v)}{\partial x} + v\frac{\partial(p)}{\partial x} = 0 \tag{B.2}$$

Using the momentum equation under its non conservative formulation and multiplying it by v, it is possible to show that

$$\rho v\frac{\partial(v)}{\partial t} + \rho v^2\frac{\partial(v)}{\partial x} = -v\frac{\partial(p)}{\partial x} \tag{B.3}$$

meaning the kinetic energy equation is expressed as

$$\frac{1}{2}\frac{\partial(\rho v^2)}{\partial t} + \frac{1}{2}\frac{\partial(\rho v^3)}{\partial x} + v\frac{\partial(p)}{\partial x} = 0 \tag{B.4}$$

Finally, adding the kinetic energy equation to the internal energy equation shows a one-to-one correspondence between derivatives enabling to build total energy back. As we have no source term, the total specific energy equation is formulated as

$$\frac{\partial(\rho(\varepsilon + \frac{1}{2}v^2))}{\partial t} + \frac{\partial(\rho(\varepsilon + \frac{1}{2}v^2)v)}{\partial x} + \frac{\partial(pv)}{\partial x} = 0 \tag{B.5}$$

Thanks to what has been proven in paragraph 2.1.2, we now know that Euler's equations are conservative for the density, momentum and the total energy.

Appendix C

Multi-fluid total energy Euler equation proof

Consider the multi-fluid momentum and specific internal energy Euler equations :

$$\frac{\partial(\alpha^k\rho^k v^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k v^k v^k)}{\partial x} + \alpha^k\frac{\partial(p^k)}{\partial x} = 0 \qquad (E_{\rho v})$$

$$\frac{\partial(\alpha^k\rho^k \varepsilon^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k \varepsilon^k v^k)}{\partial x} + p^k\left(\frac{\partial(\alpha^k)}{\partial t} + \frac{\partial(\alpha^k v^k)}{\partial x}\right) = 0 \qquad (E_{\varepsilon})$$

As for the mono-fluid case, multiply the momentum equation by v. Then, as the aim is to build total energy transport terms, note that $v^k\frac{\partial(\alpha^k\rho^k v^k)}{\partial t} = \frac{\partial(\alpha^k\rho^k v^k v^k)}{\partial t} - \alpha^k\rho^k v^k\frac{\partial(v^k)}{\partial t}$ and see that $v^k\frac{\partial(\alpha^k\rho^k v^k v^k)}{\partial x} = \frac{\partial(\alpha^k\rho^k v^k v^k v^k)}{\partial x} - \alpha^k\rho^k v^k v^k\frac{\partial(v^k)}{\partial x}$.

Replace these expressions in equation $v(E_{\rho v})$ to get the relation :

$$\frac{\partial(\alpha^k\rho^k(v^k)^2)}{\partial t} - \alpha^k\rho^k v^k\frac{\partial(v^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k(v^k)^3)}{\partial x} - \alpha^k\rho^k(v^k)^2\frac{\partial(v^k)}{\partial x} + \alpha^k v^k\frac{\partial(p^k)}{\partial x} = 0 \qquad (v(E_{\rho v}))$$

To build total energy terms, terms with specific internal energy are needed as well as some terms containing $\frac{1}{2}v^k v^k$. By concatenating both equations with the combination $(E_{\varepsilon}) + \frac{1}{2(}v(E_{\rho v}))$, some total energy transport terms appear :

$$\frac{\partial(\alpha^k\rho^k e^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k e^k v^k)}{\partial x} - \frac{1}{2}\alpha^k\rho^k v^k\frac{\partial(v^k)}{\partial t} - \frac{1}{2}\alpha^k\rho^k(v^k)^2\frac{\partial(v^k)}{\partial x} + \frac{1}{2}\alpha^k v^k\frac{\partial(p^k)}{\partial x} + p\frac{\partial(\alpha^k)}{\partial t} + p\frac{\partial(\alpha^k v^k)}{\partial x} = 0$$
$$((E_{\varepsilon}) + \tfrac{1}{2}v(E_{\rho v}))$$

Thanks to previous development, where a none conservative formulation of the multi-fluid momentum equation was derived, equation 3.14, we have that :

$$\frac{1}{2}\alpha^k\rho^k v^k\frac{\partial(v^k)}{\partial t} + \frac{1}{2}\alpha^k\rho^k(v^k)^2\frac{\partial(v^k)}{\partial x} = -\frac{1}{2}\alpha^k v^k\frac{\partial(p^k)}{\partial x}$$

One can note that there already is such a term inside the previous equation, meaning that the injection will make the fraction disappear. Moreover, an other term in the equation completes it to build a conservative term, leading to the final expression :

$$\frac{\partial(\alpha^k\rho^k e^k)}{\partial t} + \frac{\partial(\alpha^k\rho^k e^k v^k)}{\partial x} + \frac{\partial(\alpha^k v^k p^k)}{\partial x} + p^k\frac{\partial(\alpha^k)}{\partial t} = 0 \qquad (E_e)$$

Appendix D

Proof of unconditional instability for transport problem with centered space scheme

Given a function $f :]0, T[\times\mathbb{R} \to \mathbb{R}$ and a constant velocity $c > 0$, the associated $1D$ transport problem, without boundary conditions nor source term is expressed as :

$$\frac{\partial(f)}{\partial t} + c\frac{\partial(f)}{\partial x} = 0 \tag{D.1}$$

We use standard Euler forward for time stepping method, but use centered finite differences for advection term. Centered FD for a gradient is of order 2 and leads to the following scheme for the advection problem :

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0 \tag{D.2}$$

We know want to perform Von-Neumann analysis as this is a linear PDE, meaning it is a suiting framework. We set $u_j^n = \Gamma^n e^{ipj\Delta x}$ as the standard Von-Neumann analysis advise. After quite few simplifications, we get that

$$\Gamma = 1 + c\frac{\Delta t}{2\Delta x}\left(e^{ip\Delta x} - e^{-ip\Delta x}\right) \tag{D.3}$$

$$= 1 + iCsin(p\Delta x) \tag{D.4}$$

where $C = c\dfrac{\Delta t}{\Delta x}$. Von-Neumann analysis provides a criteria for a scheme to be stable i.e. for the error to be bounded at constant time and space step. This criteria is formulated as $|\Gamma| < 1$. However in our case we see that $|\Gamma| \geq 1$ meaning that the scheme is always unstable, independently of the time or space step.

In a more intuitive way, the centered scheme takes some advance on the information propagation, in the sense that it uses pieces of information that are not involved in the value it tries to compute. In some way, the scheme does not respect the behaviour of the quantity it approximates due to the global shape of the equation.

1 1D time-centered space-staggered Eulerian scheme

1.1 Action integral

Eulerian, hence with mass transport constraint, time centered, space centered on cells i except momentum on nodes $i+1/2$, first order in time and space

$$\begin{aligned}\mathcal{A} = \sum_{n,i} \Big[\Big(\tfrac{1}{2} m^n_{i+1/2} (u^n_{i+1/2})^2 - m^n_i \mathcal{E}(\rho^n_i, s^n_i) \Big) \Delta t^{n+1/2} \\ + \phi^n_i \Big(m^{n+1}_i - m^n_i + \rho^n_{i+1/2} u^n_{i+1/2} \Delta t^{n+1/2} - \rho^n_{i-1/2} u^n_{i-1/2} \Delta t^{n+1/2} \Big) \\ + \tau^n_i \Big(m^{n+1}_i s^{n+1}_i - m^n_i s^n_i + \rho^n_{i+1/2} s^n_{i+1/2} u^n_{i+1/2} \Delta t^{n+1/2} - \rho^n_{i-1/2} s^n_{i-1/2} u^n_{i-1/2} \Delta t^{n+1/2} \Big) \Big] , \end{aligned} \tag{1}$$

with cell masses, node masses, upwinding factors, upwinded node densities, and upwinded node entropies

$$m^n_i = \rho^n_i h_i \, , \tag{2a}$$
$$m^n_{i+1/2} = \tfrac{1}{2}(m^n_{i+1} + m^n_i) \, , \tag{2b}$$
$$\sigma^n_{i+1/2^\pm} = \mathrm{H}(\pm u^n_{i+1/2}) \, , \tag{2c}$$
$$\rho^n_{i+1/2} = \rho^n_i \sigma^n_{i+1/2^+} + \rho^n_{i+1} \sigma^n_{i+1/2^-} \, , \tag{2d}$$
$$\rho^n_{i+1/2} s^n_{i+1/2} = \rho^n_i s^n_i \sigma^n_{i+1/2^+} + \rho^n_{i+1} s^n_{i+1} \sigma^n_{i+1/2^-} \, . \tag{2e}$$

It may appear interesting for later derivations to define node masses as $m^n_{i+1/2} = \frac{1}{2}\rho^n_{i+1/2}(h^n_{i+1} + h^n_i)$ so as to match terms in the transport operators—this is akin to what happens in the cell centered scheme of Vazquez-Gonzalez et al. However, this definition is here inconsistent with cell masses.

1.2 Euler–Lagrange equations

Zero variation with respect to independent variables ϕ^n_i, τ^n_i, $u^n_{i+1/2}$, ρ^n_i, and $[\rho s]^n_i$ (this last combination decouples the equation on τ^n_i)

$$0 = (m^{n+1}_i - m^n_i)/\Delta t^{n+1/2} + \rho^n_{i+1/2} u^n_{i+1/2} - \rho^n_{i-1/2} u^n_{i-1/2} \, , \tag{3a}$$
$$0 = (m^{n+1}_i s^{n+1}_i - m^n_i s^n_i)/\Delta t^{n+1/2} + \rho^n_{i+1/2} s^n_{i+1/2} u^n_{i+1/2} - \rho^n_{i-1/2} s^n_{i-1/2} u^n_{i-1/2} \, , \tag{3b}$$
$$0 = m^n_{i+1/2} u^n_{i+1/2} + (\phi^n_{i+1} - \phi^n_i)(\rho^n_i \sigma^n_{i+1/2^+} + \rho^n_{i+1} \sigma^n_{i+1/2^-}) \, , \tag{3c}$$
$$\begin{aligned} 0 = \tfrac{1}{2} h_i [(u^n_{i+1/2})^2 + (u^n_{i-1/2})^2] - h_i (e^n_i + p^n_i/\rho^n_i - T^n_i s^n_i) + h_i (\phi^{n-1}_i - \phi^n_i)/\Delta t^{n+1/2} \\ + (\phi^n_i - \phi^n_{i+1}) \sigma^n_{i+1/2^+} u^n_{i+1/2} + (\phi^n_{i-1} - \phi^n_i) \sigma^n_{i-1/2^-} u^n_{i-1/2} \, , \end{aligned} \tag{3d}$$
$$\begin{aligned} 0 = h_i T^n_i + h_i (\tau^{n-1}_i - \tau^n_i)/\Delta t^{n+1/2} \\ + (\tau^n_i - \tau^n_{i+1}) \sigma^n_{i+1/2^+} u^n_{i+1/2} + (\tau^n_{i-1} - \tau^n_i) \sigma^n_{i-1/2^-} u^n_{i-1/2} \, . \end{aligned} \tag{3e}$$

1.3 Momentum equation

For simplicity, introduce upwind node densities $\rho^n_{i+1/2} = \rho^n_i \sigma^n_{i+1/2^+} + \rho^n_{i+1}\sigma^n_{i+1/2^-}$. The combination exactly eliminates ϕ

$$\frac{1}{\Delta t^{n+1/2}}\left(\frac{(3c)^n_{i+1/2}}{\rho^n_{i+1/2}} - \frac{(3c)^{n-1}_{i+1/2}}{\rho^{n-1}_{i+1/2}}\right) + \left(\frac{(3d)^n_{i+1}}{h_{i+1}} - \frac{(3d)^n_i}{h_i}\right) + \frac{\sigma^n_{i+3/2^+}u^n_{i+3/2}}{h_{i+1}\rho^n_{i+3/2}}(3c)^n_{i+3/2} + \frac{\sigma^n_{i+1/2^-}u^n_{i+1/2}}{h_{i+1}\rho^n_{i+1/2}}(3c)^n_{i+1/2} - \frac{\sigma^n_{i+1/2^+}u^n_{i+1/2}}{h_i\rho^n_{i+1/2}}(3c)^n_{i+1/2} - \frac{\sigma^n_{i-1/2^-}u^n_{i-1/2}}{h_i\rho^n_{i-1/2}}(3c)^n_{i-1/2} . \quad (4)$$

However, the result is heavy, implicit, has wide stencil, and does not ensure exact momentum conservation. It is thus appropriate to provide an approximation of (4) to the scheme order enforcing conservation and explicit transport without Debar artefact.

In order to avoid the Debar artefact, transport on node variables must match transport of node masses. Averaging (3a) at cells i and $i+1$, it is found

$$0 = (m^{n+1}_{i+1/2} - m^n_{i+1/2})/\Delta t^{n+1/2} + \tfrac{1}{2}\rho^n_{i+3/2}u^n_{i+3/2} - \tfrac{1}{2}\rho^n_{i-1/2}u^n_{i-1/2} . \quad (5)$$

The only unspecified term is now the pressure gradient which is embedded in the second term of (4). It is found to lowest order

$$(e^n_{i+1} + p^n_{i+1}/\rho^n_{i+1}) - (e^n_i + p^n_i/\rho^n_i) \approx 2(p^n_{i+1} - p^n_i)/(\rho^n_{i+1} + \rho^n_i) . \quad (6)$$

The final momentum equation is thus

$$\frac{m^{n+1}_{i+1/2}u^{n+1}_{i+1/2} - m^n_{i+1/2}u^n_{i+1/2}}{\Delta t^{n+1/2}} + \tfrac{1}{2}\rho^n_{i+3/2}(u^n_{i+3/2})^2 - \tfrac{1}{2}\rho^n_{i-1/2}(u^n_{i-1/2})^2 = -(p^n_{i+1} - p^n_i) . \quad (7)$$

1.4 Kinetic energy equation

Form the product of (7) with $\frac{1}{2}(u^{n+1}_{i+1/2} + u^n_{i+1/2})$, the kinetic energy equation is found as

$$\frac{\frac{1}{2}m^{n+1}_{i+1/2}(u^{n+1}_{i+1/2})^2 - \frac{1}{2}m^n_{i+1/2}(u^n_{i+1/2})^2}{\Delta t^{n+1/2}} + \tfrac{1}{4}\rho^n_{i+3/2}(u^n_{i+3/2})^3 - \tfrac{1}{4}\rho^n_{i-1/2}(u^n_{i-1/2})^3 + \frac{\frac{1}{2}(m^{n+1}_{i+1/2} - m^n_{i+1/2})u^{n+1}_{i+1/2}u^n_{i+1/2}}{\Delta t^{n+1/2}} + \tfrac{1}{4}\rho^n_{i+3/2}(u^n_{i+3/2})^2(u^{n+1}_{i+1/2} + u^n_{i+1/2} - u^n_{i+3/2}) - \tfrac{1}{4}\rho^n_{i-1/2}(u^n_{i-1/2})^2(u^{n+1}_{i+1/2} + u^n_{i+1/2} - u^n_{i-1/2}) = -\tfrac{1}{2}(p^n_{i+1} - p^n_i)(u^{n+1}_{i+1/2} + u^n_{i+1/2}) . \quad (8)$$

As for the momentum equation (7), the first line of the kinetic energy equation mimics the node mass equation (5). The second and third lines are thus a numerical residue—hopefully dissipative, i.e. positive—which, by substituting $m^{n+1}_{i+1/2} - m^n_{i+1/2}$ with (5), is expressed as

$$\tfrac{1}{4}\rho^n_{i+3/2}u^n_{i+3/2}(u^{n+1}_{i+1/2} - u^n_{i+3/2})(u^n_{i+3/2} - u^n_{i+1/2}) - \tfrac{1}{4}\rho^n_{i-1/2}u^n_{i-1/2}(u^{n+1}_{i+1/2} - u^n_{i-1/2})(u^n_{i-1/2} - u^n_{i+1/2}) . \quad (9)$$

This expression is close but not equal to a flux divergence and thus contains dissipation. For vanishing time step it also makes appear squared velocity gradients making sign analysis tractable. Now, up to a flux term, dissipation can be centered on cell i as

$$\begin{aligned} D_i^n &= \tfrac{1}{4}\rho_{i+1/2}^n u_{i+1/2}^n (u_{i-1/2}^{n+1} - u_{i+1/2}^n)(u_{i+1/2}^n - u_{i-1/2}^n) \\ &\qquad - \tfrac{1}{4}\rho_{i-1/2}^n u_{i-1/2}^n (u_{i+1/2}^{n+1} - u_{i-1/2}^n)(u_{i-1/2}^n - u_{i+1/2}^n) \\ &= \tfrac{1}{4\Delta t^{n+1/2}}(m_i^{n+1} - m_i^n)(u_{i+1/2}^n - u_{i-1/2}^n)^2 \\ &\qquad - \tfrac{1}{4}\big(\rho_{i-1/2}^n u_{i-1/2}^n (u_{i+1/2}^{n+1} - u_{i+1/2}^n) - \rho_{i+1/2}^n u_{i+1/2}^n (u_{i-1/2}^{n+1} - u_{i-1/2}^n)\big)(u_{i+1/2}^n - u_{i-1/2}^n)\,. \end{aligned} \tag{10}$$

In the second expression, the first part of the residue is antidissipative if $m_i^{n+1} < m_i^n$ i.e. for expansion. It can be compensated with added linear artificial viscosity. The second part vanishes with $\Delta t^{n+1/2}$.

1.5 Internal energy equation

The internal energy equation is built from the mass transport operator (3a), the pressure term in (8), and the kinetic energy dissipation (8)

$$\begin{aligned} &\frac{m_i^{n+1} e_i^{n+1} - m_i^n e_i^n}{\Delta t^{n+1/2}} + \rho_{i+1/2}^n e_{i+1/2}^n u_{i+1/2}^n - \rho_{i-1/2}^n e_{i-1/2}^n u_{i-1/2}^n \\ &\qquad = -\tfrac{1}{2} p_i^n (u_{i+1/2}^{n+1} - u_{i-1/2}^{n+1} + u_{i+1/2}^n - u_{i-1/2}^n) + D_i^n\,, \end{aligned} \tag{11}$$

with the definition

$$\rho_{i+1/2}^n e_{i+1/2}^n = \rho_i^n e_i^n \sigma_{i+1/2^+}^n + \rho_{i+1}^n e_{i+1}^n \sigma_{i+1/2^-}^n\,. \tag{12}$$

1.6 Summary of scheme step

Adding artificial viscosity, the simplest version of the scheme is

$$\rho_{i+3/2}^n \text{ see (2d)}\,, \tag{13a}$$

$$0 = \frac{m_i^{n+1} - m_i^n}{\Delta t^{n+1/2}} + \rho_{i+1/2}^n u_{i+1/2}^n - \rho_{i-1/2}^n u_{i-1/2}^n\,, \tag{13b}$$

$$p_i^n = \mathcal{P}\big(\rho_i^n, e_i^n\big)\,, \tag{13c}$$

$$q_i^n = \mathcal{Q}\big(\rho_i^n, e_i^n, \{u_{i+1/2}^n\}\big)\,, \tag{13d}$$

$$\begin{aligned} 0 = \frac{m_{i+1/2}^{n+1} u_{i+1/2}^{n+1} - m_{i+1/2}^n u_{i+1/2}^n}{\Delta t^{n+1/2}} &+ \tfrac{1}{2}\rho_{i+3/2}^n (u_{i+3/2}^n)^2 - \tfrac{1}{2}\rho_{i-1/2}^n (u_{i-1/2}^n)^2 \\ &+ (p_{i+1}^n + q_{i+1}^n - p_i^n - q_i^n)\,, \end{aligned} \tag{13e}$$

$$D_i^n \text{ see (10)}\,, \tag{13f}$$

$$\begin{aligned} 0 = \frac{m_i^{n+1} e_i^{n+1} - m_i^n e_i^n}{\Delta t^{n+1/2}} &+ \rho_{i+1/2}^n e_{i+1/2}^n u_{i+1/2}^n - \rho_{i-1/2}^n e_{i-1/2}^n u_{i-1/2}^n - D_i^n \\ &+ \tfrac{1}{2}(p_i^n + q_i^n)(u_{i+1/2}^{n+1} - u_{i-1/2}^{n+1} + u_{i+1/2}^n - u_{i-1/2}^n)\,. \end{aligned} \tag{13g}$$

This is very similar to the Kraken scheme of DeBar [1] implemented in the BBC code. Positive entropy production by artificial viscosity can be ensured with a prediction correction step on momentum—which repeats steps (13d) to (13f).

Bibliographie

[1] R.B. DeBar,
Fundamentals of the Kraken code,
Lawrence Livermore Laboratory, Report UCID-17366 (1974).

Appendix F

Proof of MacALlor total energy conservativity

In this appendix, we will prove that the scheme provided by A. Llor is conservative for total energy, even if its writing does not seem to be. We keep A. Llor's notations and we define total energy to be

$$E_i^n = m_i^n e_i^n + \frac{1}{2} m_{i+\frac{1}{2}}^n (u_{i+\frac{1}{2}}^n)^2 \tag{F.1}$$

Recall that there are N cells, meaning $N+1$ interfaces. We will assume that undefined values are null. As MacALlor scheme is based on interpolation of all values to the interfaces, it is natural to perform the discrete integration on the dual mesh i.e. to consider interface-centered cells rather than primal cells. This changes the summation to be between 0 and N rather than 1 and N.

Proving a scheme is conservative for total energy returns to proving that the total energy derivative is either null or only depends on boundary fluxes. Thus we need the total energy equation which we build thanks to the internal energy one and the kinetic energy one. A discrete integration is performed over it leading to

$$\sum_{i=0}^{N} \frac{E_i^{n+1} - E_i^n}{\Delta t^{n+\frac{1}{2}}} = -\sum_{i=0}^{N} \left(\rho_{i+\frac{1}{2}}^n e_{i+\frac{1}{2}}^n u_{i+\frac{1}{2}}^n - \rho_{i-\frac{1}{2}}^n e_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^n \right) + \sum_{i=0}^{N} D_i^n \tag{F.2}$$

$$-\frac{1}{2} \sum_{i=0}^{N} (p_i^n + q_i^n) \left(u_{i+\frac{1}{2}}^{n+1} - u_{i-\frac{1}{2}}^{n+1} + u_{i+\frac{1}{2}}^n - u_{i-\frac{1}{2}}^n \right) \tag{F.3}$$

$$-\frac{1}{4} \sum_{i=0}^{N} \left(\rho_{i+\frac{3}{2}}^n (u_{i+\frac{3}{2}}^n)^3 - \rho_{i-\frac{1}{2}}^n (u_{i-\frac{1}{2}}^n)^3 \right) \tag{F.4}$$

$$-\frac{1}{4} \sum_{i=0}^{N} \rho_{i+\frac{3}{2}}^n u_{i+\frac{3}{2}}^n \left(u_{i+\frac{1}{2}}^{n+1} - u_{i+\frac{3}{2}}^n \right) \left(u_{i+\frac{3}{2}}^n - u_{i+\frac{1}{2}}^n \right) \tag{F.5}$$

$$+\frac{1}{4} \sum_{i=0}^{N} \rho_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^n \left(u_{i+\frac{1}{2}}^{n+1} - u_{i-\frac{1}{2}}^n \right) \left(u_{i-\frac{1}{2}}^n - u_{i+\frac{1}{2}}^n \right) \tag{F.6}$$

$$-\frac{1}{2} \sum_{i=0}^{N} (p_{i+1}^n + q_{i+1}^n - p_i^n - q_i^n) \left(u_{i+\frac{1}{2}}^{n+1} + u_{i+\frac{1}{2}}^n \right) \tag{F.7}$$

The idea is now to try cancel terms and simplify this whole expression to get the remainder shape. Let us start with the green terms that we denote P.

$$
\begin{aligned}
P =& -\frac{1}{2}\sum_{i=0}^{N}\left(p_i^n+q_i^n\right)\left(u_{i+\frac{1}{2}}^{n+1}-u_{i-\frac{1}{2}}^{n+1}+u_{i+\frac{1}{2}}^{n}-u_{i-\frac{1}{2}}^{n}\right)-\frac{1}{2}\sum_{i=0}^{N}\left(p_{i+1}^n+q_{i+1}^n-p_i^n-q_i^n\right)\left(u_{i+\frac{1}{2}}^{n+1}+u_{i+\frac{1}{2}}^{n}\right)\\
=& -\frac{1}{2}\sum_{i=0}^{N}\left[\left(p_i^n+q_i^n\right)\left(u_{i+\frac{1}{2}}^{n+1}+u_{i+\frac{1}{2}}^{n}\right)-\left(p_i^n+q_i^n\right)\left(u_{i-\frac{1}{2}}^{n+1}+u_{i-\frac{1}{2}}^{n}\right)\right]\\
& -\frac{1}{2}\sum_{i=0}^{N}\left[\left(p_{i+1}^n+q_{i+1}^n\right)\left(u_{i+\frac{1}{2}}^{n+1}+u_{i+\frac{1}{2}}^{n}\right)-\left(p_i^n-q_i^n\right)\left(u_{i+\frac{1}{2}}^{n+1}+u_{i+\frac{1}{2}}^{n}\right)\right]\\
=&\frac{1}{2}\sum_{i=0}^{N}\left(p_i^n+q_i^n\right)\left(u_{i-\frac{1}{2}}^{n+1}+u_{i-\frac{1}{2}}^{n}\right)-\frac{1}{2}\sum_{i=0}^{N}\left(p_{i+1}^n+q_{i+1}^n\right)\left(u_{i+\frac{1}{2}}^{n+1}+u_{i+\frac{1}{2}}^{n}\right)\\
=&\frac{1}{2}\sum_{i=1}^{N}\left(p_i^n+q_i^n\right)\left(u_{i-\frac{1}{2}}^{n+1}+u_{i-\frac{1}{2}}^{n}\right)-\frac{1}{2}\sum_{i=0}^{N-1}\left(p_{i+1}^n+q_{i+1}^n\right)\left(u_{i+\frac{1}{2}}^{n+1}+u_{i+\frac{1}{2}}^{n}\right) \qquad \text{(discard null terms)}\\
=&\frac{1}{2}\sum_{i=0}^{N-1}\left(p_{i+1}^n+q_{i+1}^n\right)\left(u_{i+\frac{1}{2}}^{n+1}+u_{i+\frac{1}{2}}^{n}\right)-\frac{1}{2}\sum_{i=0}^{N-1}\left(p_{i+1}^n+q_{i+1}^n\right)\left(u_{i+\frac{1}{2}}^{n+1}+u_{i+\frac{1}{2}}^{n}\right) \qquad \text{(shift indices)}
\end{aligned}
$$

Then P is a vanishing term. Going to the blue term, that will be denoted Q, it can be simplified according to the following development.

$$
\begin{aligned}
Q =& \sum_{i=0}^{N}D_i^n-\frac{1}{4}\sum_{i=0}^{N}\rho_{i+\frac{3}{2}}^n u_{i+\frac{3}{2}}^n\left(u_{i+\frac{1}{2}}^{n+1}-u_{i+\frac{3}{2}}^{n}\right)\left(u_{i+\frac{3}{2}}^{n}-u_{i+\frac{1}{2}}^{n}\right)+\frac{1}{4}\sum_{i=0}^{N}\rho_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^n\left(u_{i+\frac{1}{2}}^{n+1}-u_{i-\frac{1}{2}}^{n}\right)\left(u_{i-\frac{1}{2}}^{n}-u_{i+\frac{1}{2}}^{n}\right)\\
=&\frac{1}{4}\sum_{i=0}^{N}\rho_{i+\frac{1}{2}}^n u_{i+\frac{1}{2}}^n\left(u_{i-\frac{1}{2}}^{n+1}-u_{i+\frac{1}{2}}^{n}\right)\left(u_{i+\frac{1}{2}}^{n}-u_{i-\frac{1}{2}}^{n}\right)-\frac{1}{4}\sum_{i=0}^{N}\rho_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^n\left(u_{i+\frac{1}{2}}^{n+1}-u_{i-\frac{1}{2}}^{n}\right)\left(u_{i-\frac{1}{2}}^{n}-u_{i+\frac{1}{2}}^{n}\right)\\
&-\frac{1}{4}\sum_{i=0}^{N}\rho_{i+\frac{3}{2}}^n u_{i+\frac{3}{2}}^n\left(u_{i+\frac{1}{2}}^{n+1}-u_{i+\frac{3}{2}}^{n}\right)\left(u_{i+\frac{3}{2}}^{n}-u_{i+\frac{1}{2}}^{n}\right)+\frac{1}{4}\sum_{i=0}^{N}\rho_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^n\left(u_{i+\frac{1}{2}}^{n+1}-u_{i-\frac{1}{2}}^{n}\right)\left(u_{i-\frac{1}{2}}^{n}-u_{i+\frac{1}{2}}^{n}\right)\\
=&\frac{1}{4}\sum_{i=0}^{N}\rho_{i+\frac{1}{2}}^n u_{i+\frac{1}{2}}^n\left(u_{i-\frac{1}{2}}^{n+1}-u_{i+\frac{1}{2}}^{n}\right)\left(u_{i+\frac{1}{2}}^{n}-u_{i-\frac{1}{2}}^{n}\right)-\frac{1}{4}\sum_{i=0}^{N}\rho_{i+\frac{3}{2}}^n u_{i+\frac{3}{2}}^n\left(u_{i+\frac{1}{2}}^{n+1}-u_{i+\frac{3}{2}}^{n}\right)\left(u_{i+\frac{3}{2}}^{n}-u_{i+\frac{1}{2}}^{n}\right)
\end{aligned}
$$

Now delete null terms such as $u_{-\frac{1}{2}}^n$ or $u_{N+\frac{3}{2}}^n$.

$$
\begin{aligned}
=&\frac{1}{4}\sum_{i=1}^{N}\rho_{i+\frac{1}{2}}^n u_{i+\frac{1}{2}}^n\left(u_{i-\frac{1}{2}}^{n+1}-u_{i+\frac{1}{2}}^{n}\right)\left(u_{i+\frac{1}{2}}^{n}-u_{i-\frac{1}{2}}^{n}\right)-\frac{1}{4}\sum_{i=0}^{N-1}\rho_{i+\frac{3}{2}}^n u_{i+\frac{3}{2}}^n\left(u_{i+\frac{1}{2}}^{n+1}-u_{i+\frac{3}{2}}^{n}\right)\left(u_{i+\frac{3}{2}}^{n}-u_{i+\frac{1}{2}}^{n}\right)\\
=&\frac{1}{4}\sum_{i=1}^{N}\rho_{i+\frac{1}{2}}^n u_{i+\frac{1}{2}}^n\left(u_{i-\frac{1}{2}}^{n+1}-u_{i+\frac{1}{2}}^{n}\right)\left(u_{i+\frac{1}{2}}^{n}-u_{i-\frac{1}{2}}^{n}\right)-\frac{1}{4}\sum_{i=1}^{N}\rho_{i+\frac{1}{2}}^n u_{i+\frac{1}{2}}^n\left(u_{i-\frac{1}{2}}^{n+1}-u_{i+\frac{1}{2}}^{n}\right)\left(u_{i+\frac{1}{2}}^{n}-u_{i-\frac{1}{2}}^{n}\right)
\end{aligned}
$$

Thus Q is also completly vanishing. It lasts to work on orange term denoted as R. Cancelling null terms and using telescoping principle :

$$
R=\sum_{i=1}^{N}\left(\rho_{i+\frac{1}{2}}^n e_{i+\frac{1}{2}}^n u_{i+\frac{1}{2}}^n-\rho_{i-\frac{1}{2}}^n e_{i-\frac{1}{2}}^n u_{i-\frac{1}{2}}^n\right)=\rho_{N+\frac{1}{2}}^n e_{N+\frac{1}{2}}^n u_{N+\frac{1}{2}}^n-\rho_{\frac{1}{2}}^n e_{\frac{1}{2}}^n u_{\frac{1}{2}}^n
$$

And finally purple term denoted as S which leads to this after using the same tricks as before :

$$
S=\rho_{N+\frac{1}{2}}^n\left(u_{N+\frac{1}{2}}^n\right)^3-\rho_{\frac{1}{2}}^n\left(u_{\frac{1}{2}}^n\right)^3
$$

Appendix G

Example of configuration file on Toro1.json

```
{
    "mesh": {
        "Xmin": 0.0,
        "Xmax": 1.0,
        "N": 10,
        "nb_mat": 1
    },

    "@help : scheme" : "Mac, Mac_CM, Mac_ALlor, Mac_VDF are available values",
    "scheme": "Mac",

    "linear_artificial_viscosity_coefficient": [1.0],
    "quadratic_artificial_viscosity_coefficient": [2.0],

    "@help : face_projection" : "upwind or centered",
    "face_projection" : "upwind",

    "@help : face_projection_for_momentum" : "upwind or centered. Projection for variables in momentum equation",
    "face_projection_for_momentum" : "upwind",

    "@help : dt" : "If negative, automatic time step is computed. If positive, time step remains constant.",
    "dt" : -1.0,

    "@help : power_term_type" : "scv, ncv or qcv",
    "power_term_type" : "scv",

    "@help : mixed_projection_alpha_rho_for_momentum" : "Use different densities in momentum equation
           (the time derivative related upwind, other centered).",
    "mixed_projection_alpha_rho_for_momentum" : true,

    "@help : total_energy_formulation" : "Choose to work under total energy
            formulation or not (total or specific internal one)",
    "total_energy_formulation" : true,

    "@help : pressure_projection_for_total_energy" : "upwind, centered or downwind",
    "pressure_projection_for_total_energy" : "upwind",

    "@help flux_limiter" : "minmod, superbee or umist",
    "flux_limiter" : "minmod",

    "@help : appx_order" : "1 or 2",
    "appx_order" : 1,

    "xSeparation": 0.3,

    "alpha_L": [1.0],
```

```
    "rho_L": [1.0],
    "eps_L": [2.5],
    "u_L": [0.75],

    "alpha_R": [1.0],
    "rho_R": [0.125],
    "eps_R": [2.0],
    "u_R": [0.0],

    "gamma": [1.4],
    "Pi": [0.0],
    "gravity_source_term": [0.0],

    "final_time": 0.2,
    "cfl": 0.2,

    "@help : leftBC / rightBC" : "None = boundary variable values is copy pasted
             at each time step (sort of Dirichlet condition",
    "leftBC": "None",
    "rightBC": "None",
    "@help : leftBCValue / rightBCValue": "List for [alpha, rho, v, eps] BC values",
    "leftBCValue": [
        [0.0, 0.0, 0.0, 0.0]
    ],
    "rightBCValue": [
        [0.0, 0.0, 0.0, 0.0]
    ],

    "output_file": "Results/Toro1.json",
    "output_period": 0.05,
    "conservativity_filename" : "Results/conservativity_data.txt"
}
```

Appendix H

Example of result file

To avoid adding a dozen pages to this report, result file is presented in two columns. Columns that are on a same page are consecutive : read from top left to bottom left, then top right to bottom right, then go to next page.


```
{
    "info": {
        "commit": "3e1e47074d3ed...5d09d600a7f",
        "sources_have_been_modified": true,
        "inputdata": {
            "mesh": {
                "Xmin": 0.0,
                "Xmax": 1.0,
                "N": 10,
                "nb_mat": 1
            },
            "scheme": "Mac",
            "linear_artificial_viscosity_coefficient": [0.0],
            "quadratic_artificial_viscosity_coefficient": [0.0],
            "face_projection": "upwind",
            "face_projection_for_momentum": "centered",
            "dt": 1e-05,
            "power_term_type": "scv",
            "mixed_projection_alpha_rho_for_momentum": true,
            "total_energy_formulation": false,
            "pressure_projection_for_total_energy": "upwind",
            "flux_limiter": "minmod",
            "appx_order": 2,
            "xSeparation": 0.3,
            "alpha_L": [1.0],
            "rho_L": [1.0],
            "eps_L": [2.5],
            "u_L": [0.75],
            "alpha_R": [1.0],
            "rho_R": [0.125],
            "eps_R": [2.0],
            "u_R": [0.0],
            "gamma": [1.4],
            "Pi": [0.0],
            "gravity_source_term": [0.0],
            "final_time": 0.2,
            "cfl": 0.2,
            "leftBC": "None",
            "rightBC": "None",
            "leftBCValue": [[0.0,0.0,0.0,0.0]],
            "rightBCValue": [[0.0,0.0,0.0,0.0]],
            "output_file": "Results/Toro1_Mac_scv_false_2_.json",
            "output_period": 0.2,
            "conservativity_filename": "Results/cv_data.txt"
        }
},
"results": [
    {
        "Iteration": 0,
        "Time": 0,
        "dt": 1e-05,
        "Materials": {
            "0": {
                "x": [
                    0.05,
                    0.15,
                    0.25,
                    0.35,
                    0.45,
                    0.55,
                    0.65,
                    0.75,
                    0.85,
                    0.95
                ],
                "density": [
                    1,
                    1,
                    1,
                    0.125,
                    0.125,
                    0.125,
                    0.125,
                    0.125,
                    0.125,
                    0.125
                ],
                "alpha_rho": [
                    1,
                    1,
                    1,
                    0.125,
                    0.125,
                    0.125,
                    0.125,
                    0.125,
                    0.125,
                    0.125
                ],
                "alpha": [
```

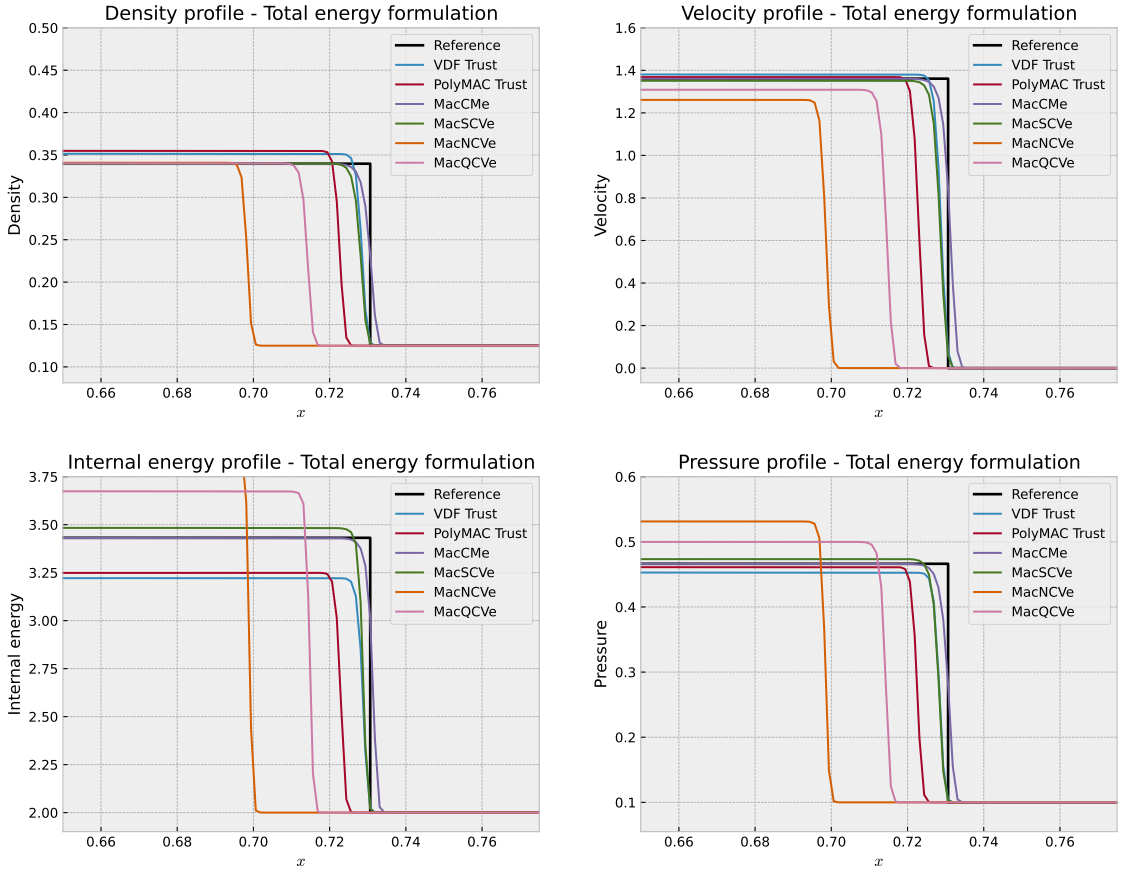
```
            1,
            1,
            1,
            1,
            1,
            1,
            1,
            1,
            1,
            1
        ],
        "eps": [
            2.5,
            2.5,
            2.5,
            2,
            2,
            2,
            2,
            2,
            2,
            2
        ],
        "pressure": [
            1,
            1,
            1,
            0.1,
            0.1,
            0.1,
            0.1,
            0.1,
            0.1,
            0.1
        ],
        "sound_speed": [
            1.18321595661992,
            1.18321595661992,
            1.18321595661992,
            1.05830052442584,
            1.05830052442584,
            1.05830052442584,
            1.05830052442584,
            1.05830052442584,
            1.05830052442584,
            1.05830052442584
        ],
        "entropy": [
            -2.22044604925031e-16,
            -2.22044604925031e-16,
            1.77635683940025e-16,
            0.608633065357724,
            0.608633065357724,
            0.608633065357724,
            0.608633065357724,
            0.608633065357724,
            0.608633065357724,
            0.608633065357724
        ],
        "Xnode": [
            0,
            0.1,
            0.2,
            0.3,
            0.4,
            0.5,
            0.6,
            0.7,
            0.8,
            0.9,
            1
        ],
        "velocity": [
            0.75,
            0.75,
            0.708333333333333,
            0.333333333333333,
            0,
            0,
            0,
            0,
            0,
            0
        ]
    }
}
},
{
    "Iteration": 20000,
    "Time": 0.2,
    "dt": 9.9999999407796e-06,
    "Materials": {
        "0": {
            "x": [
                0.05,
                0.15,
                0.25,
                0.35,
                0.45,
                0.55,
                0.65,
                0.75,
                0.85,
                0.95
            ],
            "density": [
                1,
                0.909802159078679,
                0.794914385605233,
                0.704568777946993,
                0.641394863817984,
                0.565241892621089,
                0.330122758392609,
                0.158083209268394,
                0.127418651498011,
                0.125
            ],
            "alpha_rho": [
                1,
                0.909802159078679,
                0.794914385605233,
                0.704568777946993,
                0.641394863817984,
                0.565241892621089,
                0.330122758392609,
                0.158083209268394,
                0.127418651498011,
                0.125
            ],
            "alpha": [
                1,
                1,
                1,
                1,
                1,
                1,
```

```
            1,
            1,
            1,
            1
        ],
        "eps": [
            2.5,
            2.40918378345368,
            2.23101926353533,
            2.15291356649728,
            2.36148000272658,
            2.77127867543434,
            2.9852818419054,
            2.23944818841712,
            2.01574245903956,
            2
        ],
        "pressure": [
            1,
            0.876752243121399,
            0.709387722858652,
            0.606750272228995,
            0.605856457903083,
            0.626577121393187,
            0.394203790491671,
            0.141607662646108,
            0.102737274359242,
            0.1
        ],
        "sound_speed": [
            1.18321595661992,
            1.1615261162514,
            1.11775256098109,
            1.09801256697657,
            1.14996904372548,
            1.24575922964401,
            1.29296474486624,
            1.11986203860725,
            1.06245742364678,
            1.05830052442584
        ],
        "entropy": [
            -2.22044604925031e-16,
            0.000808523484257528,
            -0.0220238375005495,
            -0.00940092959053934,
            0.120641810214333,
            0.331218701863083,
            0.620719709338183,
            0.627792256645428,
            0.608807719872706,
            0.608633065357724
        ],
        "Xnode": [
            0,
            0.1,
            0.2,
            0.3,
            0.4,
            0.5,
            0.6,
            0.7,
            0.8,
            0.9,
            1
        ],
        "velocity": [
            0.763647427754095,
            0.844450414681312,
            0.991261323687371,
            1.15115679863026,
            1.31243348598205,
            1.33828956104827,
            0.965256008344546,
            0.379248902719932,
            0.057623838096813,
            0.00207640678510553
        ]
                }
            }
        }
    ],
    "terminated_successfully": true
}
```

Appendix I

Zooms on first contact shock for first version of the schemes

Figure I.1: Toro1 test case under total energy formulation - plot of the variables of the problem at $t = 0.2$ for basic schemes



These plots confirm the conclusions made in the original paragraph that are : MacSCVe does perform very well and MacCMe seems fully conservative. In fact, on this shock, MacCMe has a logistic function profile which is centered on the shock. This is a good characteristic for high resolution computations.

Appendix J

Detailed methodology for conservativity checking

Here is shown how to compute the error for the density integral, but it could be applied with the exact same procedure for the other conserved quantities. For instance, consider E_ρ, defined as a function of time, and assume $\Omega = [0, 1]$. The theoretic part of the integral is obtained thanks to the density equation, as follows :

$$\begin{aligned} &\frac{\partial(\rho)}{\partial t} + \frac{\partial(\rho v)}{\partial x} = 0 \\ \text{i.e. } &\frac{\partial}{\partial t}\int_\Omega \rho(t,x)dx = -[\rho v(x)]_0^1 \\ \text{i.e. } &\int_\Omega \rho(t,x)dx = \int_\Omega \rho(t=0,x)dx - t[\rho v(x)]_0^1 \end{aligned}$$

Boundary values are actually initial values of left and right states meaning that we can rewrite $[\rho v(x)]_0^1$ as $[\rho_R v_R - \rho_L v_L]$ to take into account the left and right states of the Riemann problem. Using either the density equation, the momentum or the total energy one (see appendix B), such a method can be derived for all the quantities Euler's equations are supposed to preserve, leading to the theoretic term :

$$\textbf{Density :} \quad \int_0^1 \rho_{th}(t,x)dx = \int_0^1 \rho_{th}(t=0,x)dx - (\rho_R v_R - \rho_L v_L)\, t \tag{J.1}$$

$$\textbf{Momentum :} \quad \int_0^1 (\rho v)_{th}(t,x)dx = \int_0^1 (\rho v)_{th}(t=0,x)dx - \left(\rho_R v_R^2 + p_R - \rho_L v_L^2 - p_L\right) t \tag{J.2}$$

$$\begin{aligned} \textbf{Total energy :} \quad \int_0^1 \rho e_{th}(t,x)dx = &\int_0^1 \rho(\varepsilon + \frac{1}{2}v^2)_{th}(t=0,x)dx \\ &- \left(\rho_R(\varepsilon_R + \frac{1}{2}v_R^2) + p_R v_R - \rho_L(\varepsilon_L + \frac{1}{2}v_L^2) - p_L v_L\right) t \end{aligned} \tag{J.3}$$

Thank to this method, there is no need to use a solver, meaning no additional numerical noise is introduced. Thus, up to the computer's numerical precision, this method is the closest to the exact solution.

For the integration over the domain, no particular issue has to be handled for mass and total energy whereas momentum will create some. In fact, if we were to integrate over the domain to have the overall momentum, the interior would not cause any problem however the boundary would as velocity is staggered.

Figure J.1: Integration over the domain

In fact, we need to integrate over whole cells meaning that if integrating at node $\frac{1}{2}$, half the dual cell's volume would not exist. Thus, integration starts at node $\frac{3}{2}$ and ends at node $N-\frac{1}{2}$ (see grey region). However, some volume is still missing in the integral for nodes $\frac{1}{2}$ and $N+\frac{1}{2}$. Then, adding only half their volume can help correcting this missing quantity, meaning that we still integrate over the dual mesh (for momentum only) but add half their weight :

$$\int_0^1 \rho v dx \approx \sum_{i=1}^{N-1} \left(\rho_{i+\frac{1}{2}} v_{i+\frac{1}{2}}\right) + \frac{\Delta x}{2}\left(\rho_L v_L + \rho_R v_R\right) \tag{J.4}$$

Appendix K

Conservativity results for internal energy formulation

The interest in having those plots is to see the difference with total energy formulation. In fact, Euler equations do not preserve specific internal energy as internal energy can be converted in kinetic one, creating entropy.

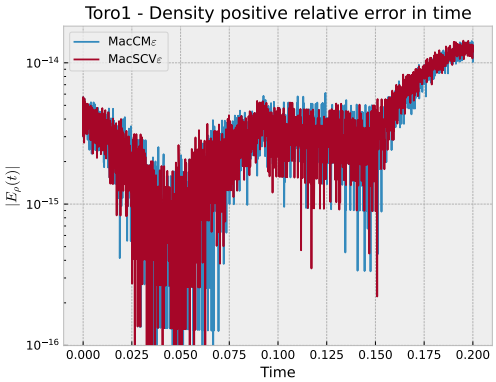


Figure K.1: Density positive relative error in time for internal energy formulation

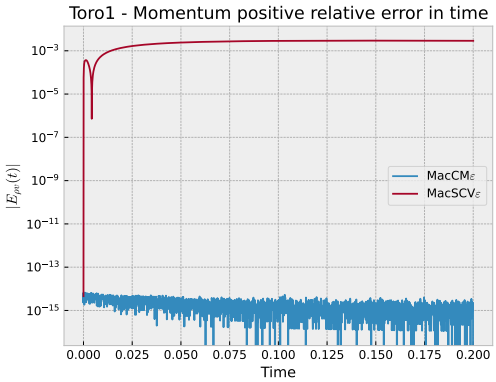


Figure K.2: Momentum positive relative error in time for internal energy formulation

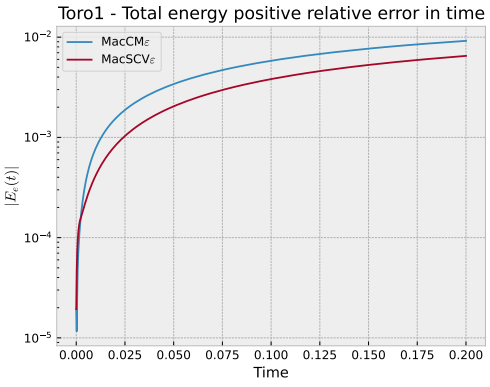


Figure K.3: Total energy positive relative error in time for internal energy formulation

We witness that density is preserved and we have the same results as before for the momentum.

Bibliography

[1] Gouvernement temporaire de la République Française. Order of creation of the French Atomic Energy Comission (CEA, Commissariat à l'Energie Atomique). *Journal officiel*, october 18th 1945. Legifrance - French national legal information website.

[2] CEA and French Military Applications Direction (DAM). *From the pioneering era to the simulation program.* CEA, 2016. History and presentation of the Military Applications Direction interests and research topics.

[3] Eleuterio Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*, pages 3, 25–30. Springer International Publishing, January 2009. Course on numerical methods for fluid dynamics with compressible and incompressible case. Pages refering to Euler's equations.

[4] Leonhard Euler. *Principes généraux du mouvement des fluides*, pages 274–315. Mémoires de l'académie des sciences de Berlin, 1757. Euler creates so-called Euler's equations, for incompressible fluids, thus only with momentum conservation equation.

[5] Henri Navier. *Sur les lois du mouvement des fluides*, volume 6, pages 389–440. Mémoires de l'Académie royale des Sciences de l'Institut de France, 1827. Navier introduces viscosity in Euler's equation.

[6] Melvin R. Baer and J. W. Nunziato. A two-phase mixture theory for the deflagration-to-detonation (ddt) transition in reactive granular materials. *Int. J Multiphase Flow*, 12(6):861–889, Frebruary 1986. Fluid and Thermal Sciences Department, Sandia National Laboratories, Albuquerque, NM 87185, U.S.A.

[7] Fabien Crouzet, Frédéric Daude, Pascal Galon, Philippe Helluy, Jean-Marc Hérard, Olivier Hurisse, and Yujie Liu. Approximate solutions to the Baer-Nunziato model. *ESAIM Proceedings*, 40:63–82, July 2013. hal-01265213.

[8] Khaled Saleh. *Analyse et Simulation Numérique par Relaxation d'écoulements Diphasiques Compressibles, contribution au Traitement des Phases Evanescentes.* PhD thesis, Pierre and Marie Curie University - Paris VI, Mathématique College (UFR 929), December 2012. tel-00761099.

[9] Alexandre Chiapolino, Richard Saurel, and Boniface Nkonga. Sharpening diffuse interfaces with compressible fluids on unstructured meshes. *Journal of computational Physics*, 340, January 2018.

[10] Francis Harlow and A. Amsden. *Fluid dynamics - An introductory text*, chapter IV.A.4.f, page 3. Los Alamos Scientific Laboratory of the university of California, la-4700 edition, January 1971. 10.2172/4762484.

[11] R. Menikoff. *Empirical EOS for solids*, volume 2 of *Shock Wave Science and Technology Reference Library*, chapter 4, pages 143–188. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. 10.1007/978-3-540-68408-4.

[12] Tore Flåtten, Alexandre Morin, and Svend Tollak Munkejord. On solution to equilibrium problems for systems of stiffened gas. *Society for Industrial and Applied Mathematics*, 71:41–67, January 2011. 10.1137/100784321.

[13] O. Le Métayer, J. Massoni, and R. Saurel. Élaboration des lois d'état d'un liquide et de sa vapeur pour les modèles d'écoulements diphasiques. *International journal of Thermal science*, 43(3):265–276, September 2003. doi:10.1016.

[14] Mark L. Wilkins. Use of artificial viscosity in multidimensional Fluid Dynamic Calculations. *Journal of computational Physics*, 36(3):281–303, July 1979. 10.1016/0021-9991(80)90161-8.

[15] Régis Debord, Pascal Galon, and Thierry Grunenwald. Modélisation d'impact par la méthode particulaire SPH, Etude de cas. January 1999. 10.13140/RG.2.1.1985.1364.

[16] Francis H. Harlow and J. Eddie Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *The Physics of Fluids*, 8(12):2182–2189, December 1965. 10.1063/1.1761178.

[17] I.K. Park, H.K. Cho, H.Y. Yoon, and J.J. Jeong. Numerical effects of the semi-conservative form of momentum equations for multi-dimensional two-phase flows. *Nuclear Engineering and Design*, 239(11):2365–2371, 2009. 10.1016/j.nucengdes.2009.06.011.

[18] United States Nuclear Energy Regulatory Commission. TRACE V5.0 - THEORY MANUAL - Field Equations, Solution Methods and Physical Models. 2000. Manual for underlying theory.

[19] R. B. DeBar. Fundamentals of the kraken code. [eulerian hydrodynamics code for compressible nonviscous flow of several fluids in two-dimensional (axially symmetric) region]. *Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States)*, (UCID-17366), March 1974. Technical Report.

[20] Antoine Llor and Thibaud Vazquez-Gonzalez. Geometry, energy, and entropy compatible (geec) variational approaches to various numerical schemes for fluid dynamics. In Frank Nielsen and Frédéric Barbaresco, editors, *Geometric Science of Information*, pages 559–567, Cham, 2019. Springer International Publishing.

[21] Medi Bijan and Mohammad Amanullah. Application of a Finite-Volume Method in the Simulation of Chromatographic Systems: Effects of Flux Limiters. *Industrial and Engineering Chemistry Research Journal*, 50(50):1739–1748, January 2011. 10.1021/ie100617c.

[22] Python website, Floating point arithmetic : errors and limitations.