Simulation of two-phase flows by domain decomposition

Thu Huyên Dao

École centrale des arts et manufactures
Mathématiques Appliques aux systèmes

Thèse

présentée en première version en vu d'obtenir le grade de Docteur,
spécialité « Mathématiques appliquées »

par

DAO Thu Huyen

Simulation numérique d'écoulements diphasiques par décomposition de domaine

À mes parents

Titre Simulation numérique d'écoulements diphasiques par décomposition de domaine

Résumé Ce travail a été consacré à la simulation numérique des équations de la mécanique des fluides par des méthodes de volumes finis implicites.
Tout d'abord, nous avons étudié et mis en place une version implicite du schéma de Roe pour les écoulements monophasiques et diphasiques compressibles. Grâce à la méthode de Newton utilisée pour résoudre les systèmes nonlinéaires, nos schémas sont conservatifs. Malheureusement, la résolution de ces systèmes est très coûteuse. Il est donc impératif d'utiliser des algorithmes de résolution performants. Pour des matrices de grande taille, on utilise souvent des méthodes itératives dont la convergence dépend de leur spectre. Nous avons donc étudié le spectre du système linéaire et proposé une stratégie de Scaling pour améliorer le conditionnement de la matrice. Combinée avec le préconditionneur classique ILU, notre stratégie de Scaling a réduit de façon significative le nombre d'itérations GMRES du système local et le temps de calcul. Nous avons également montré l'intérêt du schéma centré pour la simulation de certains écoulements à faible nombre de Mach.
Nous avons ensuite étudié et implémenté la méthode de décompostion de domaine pour les écoulements compressibles. Nous avons proposé une nouvelle variable interface qui rend la méthode du complément de Schur plus facile à construire et nous permet de traiter les termes de diffusion. L'utilisation du solveur itératif GMRES plutôt que Richardson pour le système interface apporte aussi une amélioration des performances par rapport aux autres méthodes. Nous pouvons également découper notre domaine de calcul en un nombre quelconque de sous-domaines. En utilisant la stratégie de Scaling pour le système interface, nous avons amélioré le conditionnement de la matrice et réduit le nombre d'itérations GMRES de ce système. En comparaison avec le calcul distribué classique, nous avons montré que notre méthode est robuste et efficace.

Mots-clés Equations d'Euler - Equations de Navier-Stokes - Modèle bifluide - Volumes finis - Schéma de Roe - Schéma implicite - Méthode de Newton - Préconditionneur - Décomposition de domaine - Complément de Schur - Calcul parallèle

Title Simulation of two-phase flows by domain decomposition

Abstract This thesis deals with numerical simulations of compressible fluid flows by implicit finite volume methods.
Firstly, we studied and implemented an implicit version of the Roe scheme for compressible single-phase and two-phase flows. Thanks to Newton method for solving nonlinear systems, our schemes are conservative. Unfortunately, the resolution of nonlinear systems is very expensive. It is therefore essential to use an efficient algorithm to solve these systems. For large size matrices, we often use iterative methods whose convergence depends on the spectrum. We have studied the spectrum of the linear system and proposed a strategy, called Scaling, to improve the condition number of the matrix. Combined with the classical ILU preconditioner, our strategy has reduced significantly the GMRES iterations for local systems and the computation time. We also show some satisfactory results for low Mach-number flows using the implicit centered scheme.
We then studied and implemented a domain decomposition method for compressible fluid flows. We have proposed a new interface variable which makes the Schur complement method easy to build and allows us to treat diffusion terms. Using GMRES iterative solver rather than Richardson for the interface system also provides a better performance compared to other methods. We can also decompose the computational domain into any number of subdomains. Moreover, the Scaling strategy for the interface system has improved the condition number of the matrix and reduced the number of GMRES iterations. In comparison with the classical distributed computing, we have shown that our method is more robust and efficient.

Keywords Euler equations - Navier-Stokes equations - Two-fluid model - Finite volume methods- Roe scheme - Implicit scheme - Newton method - Preconditioner - Domain decomposition - Schur complement - Parallel computing

Table des matières

Liste des figures

Introduction générale

La recherche d'un meilleur rendement et d'une sûreté optimale des centrales nucléaires en France est depuis longtemps un enjeu industriel majeur. La modélisation et la simulation numérique jouent à ce titre un rôle clé. Elles permettent la compréhension et la prédiction des phénomènes physiques mis en jeu au sein d'un réacteur nucléaire.

Le parc nucléaire français est composé en grande partie de réacteurs à eau pressurisée. Le réacteur peut se diviser en trois parties principales : le circuit primaire, le circuit secondaire et le circuit de refroidissement. Le circuit primaire est associé au processus de chauffage du fluide caloporteur qui est, dans ce cas, l'eau. Il s'agit d'un circuit fermé qui se situe dans une enceinte de confinement. On y retrouve la cuve qui contient le coeur du réacteur nucléaire où la réaction en chaîne se produit. Le combustible nucléaire est assemblé sous forme de crayons et des barres de contrôle permettent de gérer le taux de réaction nucléaire. L'eau présente est chauffée à une température de l'ordre de 300°C mais reste sous forme liquide grâce à un pressuriseur assurant une pression dans le circuit d'environ 155 bars. Cette eau chaude circule ensuite dans le générateur de vapeur permettant l'échange de la chaleur emmagasinée avec de l'eau froide aux pressions de 75 bars. Le générateur de vapeur fait partie du circuit secondaire. L'eau froide qui y est présente est donc chauffée jusqu'à l'état vapeur. Cette vapeur chaude va alors entraîner des turbines réliées à des alternateurs et l'energie mécanique est ainsi transformée en électricité. Enfin, la même vapeur est recondensée en eau liquide par contact avec l'eau du circuit de refroidissement puis recircule dans le générateur.

Nous voyons donc, par cette description simplifiée, que de nombreux phénomènes physiques entrent en jeu au cours du processus de production d'électricité. Des écoulements diphasiques eau-vapeur évoluent dans quasiment toutes les parties de la centrale.

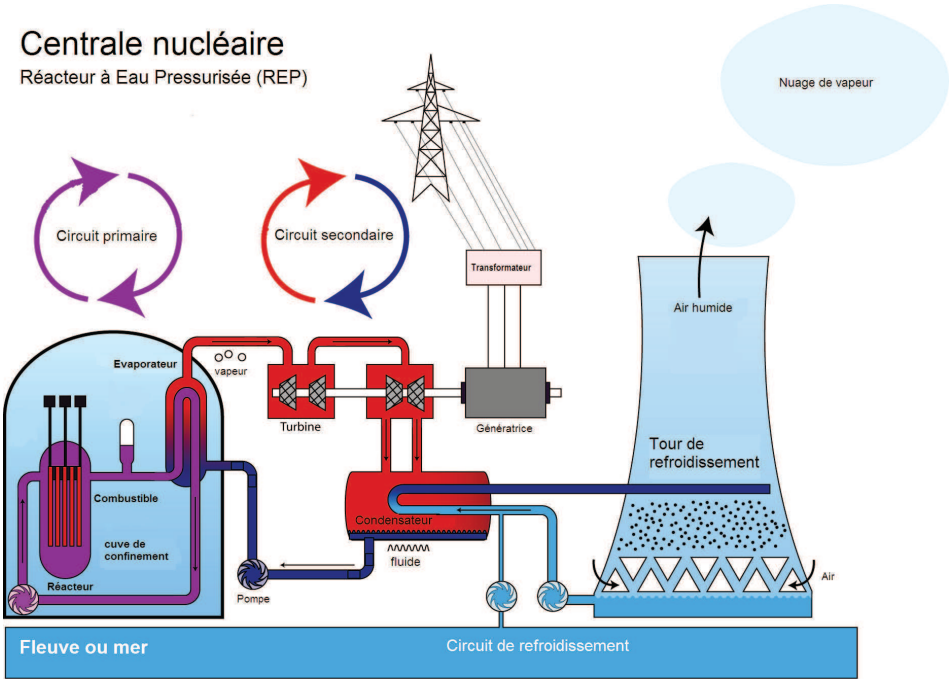

FIGURE 1 – *Schéma simplifié d'un réacteur nucléaire à eau pressurisée*

Au Laboratoire de Modélisation et de simulation à l'Échelle Composant (LMEC) du Commissariat à l'Énergie Atomique et aux Alternative de Saclay, où j'ai effectué ma thèse, plusieurs outils informatiques sont développés et améliorés dans le but de simuler ces écoulements. Parmis les logiciels scientifiques disponibles, on peut tout d'abord citer FLICA-4 Toumi et al. (2000), utilisé principalement pour la simulation des écoulements diphasiques dans le coeur du réacteur nucléaire. Le modèle mathématique utilisé dans FLICA-4 est un modèle de mélange à quatre équations en milieux poreux. Un modèle de mélange , comme son nom l'indique, représente l'évolution du fluide grâce à des équations de conservation écrites sur les grandeurs physiques associées au mélange : la masse, la quantité de mouvement et l'énergie. Le modèle est complété par des fermetures algébriques exprimant l'équilibre cinématique entre ces deux phases. Un autre code en cours de développement est FLICA-OVAP Fillion et al. (2009). Son architecture est orientée objet et dédiée à la simulation de plusieurs modèles. Différents modèles de mélange sont implémentés ainsi que des modèles multichamps qui permettent de décrire plus finement la dynamique des écoulements à plusieurs phases via des lois de conservation sur les grandeurs physiques. Parmis les modèles, le modèle dit bifluide à une pression est le plus utilisé pour la représentation des écoulements diphasiques eau-vapeur. Contrairement au modèle de mélange de FLICA-4, tout les déséquilibres entre les phases sont pris en compte. Tous les efforts visent à développer ce logiciel avec des solveurs numériques et des modèles physiques permettant de répondre aux besoins de simulation thermodydraulique diphasique des coeurs pour les échelles de modélisation "3D-poreuse" ou locale.

Mes travaux de thèse ont été financés par le CEA. Le contexte de ma thèse est celui de la simulation réaliste d'écoulements multiphasiques

par des modèles 3D instationnaires, résolus sur des architectures massivement parallèles (sur des machines du Centre de Calcul Recherche et Technologie-CCRT). Plus précisément, l'objectif est d'améliorer le speed-up (facteur d'accélération) des calculs réalisés avec le code FLICA-OVAP sur des machines multiprocesseurs afin d'obtenir des temps de calcul raisonnables sur des géométries aussi complexes que celle d'un coeur de réacteur. La formulation implicite des schémas de calcul permet l'emploi de pas de temps élévés, mais demande à chaque pas de temps la résolution d'un système linéaire $AX = b$ d'autant plus mal conditionné que le nombre de mailles et le pas de temps sont grands. Ces systèmes doivent donc être préconditionés afin de permettre une résolution précise et une augmentation modérée du temps de calcul avec la taille du système.

Lorsque le système est de grande taille, la résolution parallèle sur plusieurs processeurs devient essentielle pour obtenir des temps de calcul raisonnables. Actuellement la matrice et le second membre sont distribués sur plusieurs processeurs et on résout le système en parallèle avec un algorithme de type GMRES. Malheureusement les préconditionneurs classiquement utilisés pour la résolution en parallèle ont des performances décevantes à cause des coûts de communication et certains présentent des problèmes de robustesse. Des tests ont été réalisés sur différents cas tests de FLICA-OVAP et aboutissent à la sévère conclusion qu'il vaut mieux dans certains cas 3D ne pas utiliser les préconditionneurs en parallèle. La puissance de calcul disponible n'est alors pas utilisée de manière optimale.

Ces différents points motivent l'emploi d'une approche de résolution parallèle alternative qui s'adapte mieux aux calculs parallèlles que la distribution classique sur plusieurs processeurs. Nous avons choisi d'étudier la méthode de décomposition de domaine. Elle représente une voie naturelle et efficace vers le parallélisme. Elle permet également de traiter facilement des géométries complexes en associant plusieurs modèles ou schémas de calcul.

L'idée consiste à ramener le problème global à un ensemble de problèmes locaux, auxquels s'ajoute une condition de raccord qui assure le recollement de la solution aux interfaces de la décomposition. L'idée d'utiliser une méthode itérative remonte à Schwarz (1869). On se donne tout d'abord des conditions aux limites arbitraires aux interfaces. On est alors en mesure de résoudre les problèmes locaux. La solution calculée dans un sous-domaine permet de mettre à jour les conditions aux limites pour les sous-domaines voisins. On passe ensuite à l'iteration suivante. Cet algorithme est naturellement parallélisable, puisque les résolutions locales à chaque itération sont totelement découplées.

L'objectif de cette thèse est d'étudier et implémenter la méthode de decomposition de domaine dans une maquette d'un code C++ proche de FLICA-OVAP mais avec des modèles plus simples comme les équations de Navier-Stokes compressibles et modèle bifluide isentropiques. Cette maquette doit être suffisamment souple et évolutive et posséder une interface public que ICoCo (Interface for Code Coupling Perdu (2006))

permettant le couplage de plusieurs instances du code. Elle a pour but d'être utilisée pour étudier de façon expérimentale les propriétés de la méthode et effectuer une comparaison entre le calcul distribué et la méthode de décomposition de domaine. Elle est appelée ParaFlow et tous les résultats dans cette thèse sauf le problème de backward-facing step ont été réalisés en utilisant cette maquette.

Déroulement de la thèse

- Dans un premier temps nous avons étudié et implémenté la méthode de décompostion de domaine pour les écoulements monophasiques (équations de Navier-Stokes compressibles). Nous avons proposé une nouvelle variable interface (Dao et al. (2011a)) inspirée par l'approche présentée dans Dolean and Lanteri (2001). Notre variable interface rend la méthode du complément de Schur plus facile à construire que celle utilisée dans Dolean and Lanteri (2001). De plus, elle permet de traiter les termes de diffusion. Nos schémas sont conservatifs grâce à la méthode de Newton utilisée pour résoudre les systèmes non linéaires. L'utilisation du solveur itératif GMRES plutôt que Richardson pour le système interface apporte aussi une amélioration des performances par rapport à Dolean and Lanteri (2001). Nous pouvons également découper notre domaine de calcul en un nombre quelconque de sous-domaines.

  Nous avons aussi étudié le spectre du système linéaire et proposé la stratégie de Scaling (Dao et al. (2011b)) pour améliorer le conditionnement de la matrice A. Cette stratégie est une transformation similaire de la matrice (en utilisant son spectre) pour que les coefficients en dehors de la diagonale aient le même d'ordre de grandeur. Combinée avec le préconditionneur classique ILU, notre stratégie de Scaling a réduit de façon significative le nombre d'itération GMRES du système local et le temps de calcul. Les résultats numériques ont aussi montré que cette stratégie a aidé à améliorer le conditionnement de la matrice du système interface et à réduire le nombre d'itérations GMRES de ce système.

- Dans un deuxième temps, notre travail de thèse a porté sur la simulation numérique des écoulements diphasiques (Dao et al. (2012a), Dao et al. (2012b)). Nous avons étudié et implémenté les schémas numériques de type Roe pour le modèle bi-fluide. Nous avons également proposé une correction entropique pour rendre le schéma de Roe positif. Des tests classiques (comme le tube à choc, la sédimentation etc ..) ont été réalisés pour montrer le bon comportement de ce schéma. Nous avons également étudié le comportement asymptotique du modèle bi-fluide dans le cas où les deux phases sont supposées incompressibles. Cela donne un système de deux équations aux dérivées partielles du premiers ordre dont il est plus simple à étudier l'hyperbolicité ainsi que le comportement lorque un des phases disparaît.

Nous avons ensuite appliqué la méthode de décomposition de domaine au modèle bi-fluide et étudié son comportement dans le cas diphasique.

– Suite au succès de la stratégie de Scaling pour les écoulements monophasiques, cette stratégie a été appliquée au cas diphasique et elle a amélioré les performances.

1 Hyperbolic Systems of Conservation Laws

Contents

In this chapter, we briefly describe the basic ingredients of the theory of hyperbolic systems of conservation laws. These ingredients allow us to better understand the mathematical properties of these systems. We then describe the finite volume method for hyperbolic systems. We are especially interested in the Roe approximate Riemann solver. Different schemes with different orders are also presented. As usual with implicit schemes, at each time step, we need to solve a nonlinear system. To keep the conservation property of our finite volume scheme, Newton method is used to solve the nonlinear system. We end the chapter by describing the boundary conditions of the problem and presenting some entropy fix for the Roe scheme. More detailed description of these systems and methods can be seen in Godlewski and Raviart (1996) and LeVeque (2002).

1.1 Hyperbolic Systems

Let us consider a system of partial differential equations (PDEs), arising from the modeling of the conserved quantities,

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{i=1}^{d} \frac{\partial}{\partial x_i} \mathbf{F}_i(\mathbf{U}) = 0,\ \mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d,\ t > 0, \tag{1.1}$$

where d is the space dimension, $\mathbf{U} : (\mathbf{x}, t) \in \mathbb{R}^d \times \mathbb{R} \longmapsto \mathbf{U}(\mathbf{x}, t) \in \mathbb{R}^p$ is the unknown vector of **conserved quatities**, p is the number of unknowns. The flux functions $\mathbf{F}_i : \mathbf{U} \in \mathbb{R}^p \longmapsto \mathbf{F}_i(\mathbf{U}) \in \mathbb{R}^p, 1 \leq i \leq d$ are differentiable with respect to the state vector $\mathbf{U}$. In general, the flux functions are nonlinear functions of $\mathbf{U}$. The fundamental characteristic of the system (1.1) is the conservation of each component of $\mathbf{U}$.
One says that the system (1.1) is written in **conservation form**. The system (1.1) can be also written in the **non conservation form** or **quasilinear form** as

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{i=1}^{d} \frac{\partial \mathbf{F}_i(\mathbf{U})}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial x_i} = 0, \tag{1.2}$$

or :

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{i=1}^{d} A_i(\mathbf{U}) \frac{\partial \mathbf{U}}{\partial x_i} = 0. \tag{1.3}$$

For better understanding of the system (1.3), we recall the following basic definitions and properties :

Definition 1.1 *The system (1.3) is called hyperbolic provided that the matrix $A(\mathbf{U}, \omega) = \sum_{i=1}^{d} A_i(\mathbf{U})\omega_i$ is diagonalizable in $\mathbb{R}$ i.e $A(\mathbf{U}, \omega)$ has p real eigenvalues*

$$\lambda_1(\mathbf{U}, \omega) \leq \lambda_2(\mathbf{U}, \omega) \leq \ldots \leq \lambda_k(\mathbf{U}, \omega) \leq \ldots \leq \lambda_p(\mathbf{U}, \omega),\ \lambda_k(\mathbf{U}, \omega) \in \mathbb{R},$$

and p linearly independent corresponding eigenvectors $r_k \in \mathbb{R}^p$ and :

$$A(\mathbf{U}, \omega) \cdot r_k(\mathbf{U}, \omega) = \lambda_k(\mathbf{U}, \omega) r_k(\mathbf{U}, \omega).$$

In addition, the system (1.3) is called strictly hyperbolic if the eigenvalues of $A(\mathbf{U}, \omega)$ are all distinct for any unitary vector $\omega = (\omega_1, \cdots, \omega_d)$.

Property 1.1 *The hyperbolicity property is invariant by the change of variables.*

Definition 1.2 *The system (1.3) is called symmetrizable if there exists a matrix $A_0(\mathbf{U})$ symmetric positive definite such that the matrices*

$$A_0(\mathbf{U}) A_i(\mathbf{U}),\ 1 \leq i \leq d$$

are symmetric.

In fact, little is known about systems in more than one space variable unless they are symmetrizable. Fortunately, most of the systems of conservation laws arising in practice are symmetrizable ; this is a consequence of the existence of an entropy function. And symmetrizable systems of conservation laws are clearly hyperbolic (see Godlewski and Raviart (1996)).

Cauchy Problem For hyperbolic systems, the most important problem is the initial value problem (IVP), i.e **the Cauchy problem** : Find a function $\mathbf{U}(\mathbf{x},t) \in \mathbb{R}^d x[0,\infty) \longmapsto \mathbf{U}(\mathbf{x},t)$ which is the solution of

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{i=1}^{d} \frac{\partial}{\partial x_i} \mathbf{F}_i(\mathbf{U}) = 0, \tag{1.4}$$

$$\mathbf{U}(\mathbf{x}, t=0) = \mathbf{U}_0(\mathbf{x}), \tag{1.5}$$

where $\mathbf{U}_0(\mathbf{x}) : \mathbb{R}^d \longmapsto \mathbb{R}^d$ is a given initial condition of the vector $\mathbf{U}$.
In one-dimensional space, if the function $\mathbf{U}_0$ has the following particular form,

$$\mathbf{U}_0 = \begin{cases} \mathbf{U}_l, & \mathbf{x} < 0 \\ \mathbf{U}_r, & \mathbf{x} > 0 \end{cases} \tag{1.6}$$

the Cauchy problem is called **the Riemann problem**.

1.2 Weak Solutions

A function $\mathbf{U} : \mathbb{R}^d \times \mathbb{R}^+ \longmapsto \mathbb{R}^p$ is called a **classical solution** of the Cauchy problem (1.4), (1.5) if $\mathbf{U} \in C^1$ and satisfies the equations (1.4)-(1.5) pointwise. Unfortunately, a fundamental feature of nonlinear conservation laws is that discontinuous solutions can be easily obtained even from smooth initial data. So the classical solution is not a convenient notion to study the solutions of (1.4)-(1.5). This leads us to introducing weak solutions (solutions in the sense of distributions) of the Cauchy problem as follows :

Definition 1.3 *A mesurable and locally bounded function* $\mathbf{U} : \mathbb{R}^d \times \mathbb{R}^+ \longmapsto \mathbb{R}^p$ *is called a* **weak solution** *of the Cauchy problem (1.4), (1.5) if for any function* $\phi \in C^1$ *having compact support in* $\mathbb{R}^d \times \mathbb{R}^+$ *we have :*

$$\int_0^\infty \int_{\mathbb{R}^d} \left(\frac{\partial \phi}{\partial t} \cdot \mathbf{U} + \sum_{j=1}^{d} F_j(\mathbf{U}) \cdot \frac{\partial \phi}{\partial x_j} \right) d\mathbf{x}dt + \int_{\mathbb{R}^d} \phi(\mathbf{x},0)\mathbf{U}_0(\mathbf{x})d\mathbf{x} = 0. \tag{1.7}$$

By construction, any classical solution of the problem (1.4)-(1.5) is also a weak solution. Conversely, we have :

Proposition 1.1 *If* $\mathbf{U}_0 \in C^1$ *and* $\mathbf{U} \in L^\infty_{loc}(\mathbb{R}^d \times \mathbb{R}^+)^p$ *is a weak solution of the Cauchy problem (1.4)-(1.5) and moreover* $\mathbf{U} \in C^1$, *then* $\mathbf{U}(\bullet,\bullet)$ *is a classical solution of the Cauchy problem (1.4)-(1.5).*

Considering now the case where the function $\mathbf{U}(\bullet,\bullet)$ is smooth except on the smooth orientable surfaces Σ. We say that a function $\mathbf{U}(\bullet,\bullet)$ is "piecewise C^1" if there exists a finite number of smooth orientable surfaces Σ such that the function $\mathbf{U}(\bullet,\bullet)$ is a C^1 function in $(\mathbb{R}^d \times [0,\infty[)$ and across the surfaces Σ, $\mathbf{U}(\bullet,\bullet)$ has a jump discontinuity i.e the function $\mathbf{U}(\bullet,\bullet)$ has a limit on the left and on the right at any point $(\mathbf{x},t) \in \Sigma$. In one-dimensional space, if the curves Σ are parametrized by $x = \xi(t)$, where $\xi(t)$ is a function of class C^1, we have

$$\mathbf{U}_+(\xi(t),t) = \lim_{\varepsilon \to 0^+} \mathbf{U}(\xi(t)+\varepsilon, t), \qquad (\xi(t),t) \in \Sigma$$

$$\mathbf{U}_-(\xi(t),t) = \lim_{\varepsilon\to 0^+} \mathbf{U}(\xi(t)-\varepsilon,t), \qquad (\xi(t),t)\in\Sigma.$$

We now show that even in the frame of "piecewise C^1"functions not every discontinuity is admissible. The admissible functions have to satisfy the so called **Rankine-Hugoniot** relation :

Theorem 1.1 *Let $\mathbf{F}\in C^1$ and the initial condition $\mathbf{U}_0$ be a piecewise C^1 function. Then a mesurable and locally bounded function $\mathbf{U}$ in the class of piecewise C^1 functions is a weak solution of the Cauchy problem (1.4)-(1.5) if and only if*

- *$\mathbf{U}$ is a classical solution of (1.4)-(1.5) in the domain $\mathbb{R}^d\times[0,\infty[$ where $\mathbf{U}$ is C^1.*
- *$\mathbf{U}$ satisfies the Rankine-Hugoniot relation :*

$$(\mathbf{U}_+-\mathbf{U}_-)n_t + \sum_{j=1}^{d}(F_j(\mathbf{U}_+)-F_j(\mathbf{U}_-))n_{x_j} = 0 \, on\, \Sigma \qquad (1.8)$$

along the surfaces of discontinuity, where $\mathbf{n}=(n_t,n_{x_1},n_{x_2},\ldots,n_{x_d})$ is the exterior normal vector of Σ.

1.3 Mathematical Entropy and Entropy Solution

However, when considering the solution in the sense of distributions, we lose the uniqueness of the solution. It is thus necessary to introduce a criterion that enables us to choose the "physicaly revelant" solution among all the weak solutions of (1.4)-(1.5). This criterion is based on the concept of entropy.

Definition 1.4 *Assume that Ω is convex. Then, a convex function $S:\mathbf{U}\in\Omega\to\mathbb{R}$ is called an entropy of the system (1.1) provided that there exists d functions $G_j:\Omega\to\mathbb{R}$, $1\le j\le d$ called entropy fluxes, such that :*

$$S'(\mathbf{U})F'_j(\mathbf{U}) = G'_j(\mathbf{U}), \qquad 1\le j\le d$$

where

$$S' = \left(\frac{\partial S}{\partial U_1},\ldots,\frac{\partial S}{\partial U_p}\right), \quad G'_j = \left(\frac{\partial G_j}{\partial U_1},\ldots,\frac{\partial G_j}{\partial U_p}\right), \quad F'_j = \left(\frac{\partial F_{ij}}{\partial U_k}\right) \quad 1\le j,k\le p.$$

Definition 1.5 *A measurable and locally bounded function $\mathbf{U}:\mathbb{R}^d\times\mathbb{R}^+\longmapsto\mathbb{R}^p$ is called a* **weak entropy solution** *of the Cauchy problem (1.4)-(1.5) if for any function $\phi\in C^1$ having compact support in $\mathbb{R}^d\times\mathbb{R}^+$, $\phi\ge 0$, we have :*

$$\int_0^\infty\int_{\mathbb{R}^d}\left(\frac{\partial\phi}{\partial t}\cdot S(\mathbf{U}) + \sum_{j=1}^{d}F_j(\mathbf{U})\cdot\frac{\partial\phi}{\partial x_j}\right)d\mathbf{x}dt + \int_{\mathbb{R}^d}\phi(\mathbf{x},0)S(\mathbf{U}_0(\mathbf{x}))d\mathbf{x}\ge 0. \qquad (1.9)$$

Theorem 1.2 *Let $\mathbf{F}\in C^1$ and let the initial condition $\mathbf{U}_0$ be piecewise C^1. A measurable and locally bounded function $\mathbf{U}$ in the class of piecewise C^1 is called a weak entropy solution of the Cauchy problem (1.4), (1.5) if and only if :*

- *$\mathbf{U}$ is the classical solution of (1.4)-(1.5) in the domains $\mathbb{R}^d\times[0,\infty[$ where $\mathbf{U}$ is a C^1 function*
- *$\mathbf{U}$ satisfies the Rankine-Hugoniot relation (1.8)*

- **U** *satisfies the following inequality*

$$n_t[S(\mathbf{U})] + \sum_{j=1}^{d}[G_j(\mathbf{U})]n_{x_j} \leq 0. \tag{1.10}$$

1.4 Numerical Schemes for Hyperbolic Systems

1.4.1 Finite Volume Method

As we saw in the previous sections, the conservative form (1.1) enables us to define the concept of weak solutions, which can be discontinuous ones. Discontinuous solutions such as shock waves are of great importance in transient calculations. In order to correctly capture shock waves, one needs a robust, low diffusive conservative scheme. The finite volume framework is the most appropriate setup to write discrete equations that express the conservation laws at each cell. For finite volume method, we decompose the computational domain Ω into N disjoint cells C_i with volume v_i. Two neighboring cells C_i and C_j have a common boundary ∂C_{ij} with area s_{ij}. We denote $N(i)$ the set of neighbors of a given cell C_i and $\mathbf{n}_{ij}$ the exterior unit normal vector of ∂C_{ij} . On C_i, $\mathbf{U}(.,t)$ is approximated by a constant $\mathbf{U}_i(t)$ which means

$$\mathbf{U}_i(t) \cong \frac{1}{v_i}\int_{C_i}\mathbf{U}(\mathbf{x},t)\mathrm{d}\mathbf{x}.$$

Integrating the system (1.1) over C_i gives

$$\frac{1}{v_i}\int_{C_i}\left(\frac{\partial \mathbf{U}}{\partial t} + \nabla\cdot(\mathbf{F}(\mathbf{U}))\right)d\mathbf{U} = 0. \tag{1.11}$$

Applying the Green-Ostrogradski theorem to the formula (1.11), we have

$$\frac{\partial}{\partial t}\left(\frac{1}{v_i}\int_{C_i}\mathbf{U}(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\right) + \frac{1}{v_i}\sum_{j\in N(i)}\int_{\partial C_{ij}}\mathbf{F}(\mathbf{U}).\mathbf{n}_{ij}\,\mathrm{d}s = 0. \tag{1.12}$$

The first term of (1.12) is naturally approximated by :

$$\frac{\partial}{\partial t}\left(\frac{1}{v_i}\int_{C_i}\mathbf{U}(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\right) \cong \frac{\partial \mathbf{U}_i(t)}{\partial t}.$$

We then denote $\mathbf{U}_i^n$ the numerical approximation over the cell C_i at n^{th} time step and we obtain

$$\frac{\partial}{\partial t}\left(\frac{1}{v_i}\int_{C_i}\mathbf{U}(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\right) \cong \frac{\mathbf{U}_i^{n+1}-\mathbf{U}_i^n}{\Delta t}.$$

Since the approximation is not continuous across ∂C_{ij}, we have to discretize $\int_{\partial C_{ij}}\mathbf{F}(\mathbf{U}).\mathbf{n}_{ij}\,\mathrm{d}s$, which represents the flux across the boundary ∂C_{ij}. The problem is then to define the numerical flux approximating $\int_{\partial C_{ij}}\mathbf{F}(\mathbf{U}).\mathbf{n}_{ij}\,\mathrm{d}s$, using only the values $\mathbf{U}_i(t)$. The usual way consists in introducing a numerical flux function Φ_{ij} such that

$$\Phi_{ij} \cong \frac{1}{s_{ij}}\int_{\partial C_{ij}}\mathbf{F}(\mathbf{U}).\mathbf{n}_{ij}\,\mathrm{d}s.$$

Φ_{ij} is a function of $\mathbf{U}_i^n$ or $\mathbf{U}_i^{n+1}$ that approximates the interfacial fluxes. We obtain the numerical scheme for system (1.1)

$$\frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i^n}{\Delta t} + \sum_{j \in N(i)} \frac{1}{v_i} \Phi_{ij} s_{ij} = 0. \tag{1.13}$$

So what are the desirable properties of such fluxes ? First of all, the obtained finite volume scheme needs to be stable. Unfortunately, the stability of the scheme is not enough to ensure the quality of the solution. The Lax-Wendroff scheme which is stable gives rise to an oscillatory solution in the neighborhood of strong gradients. So we need to look for schemes which respect the "profile" of the solution.
In the scalar case, A.Harten (1984) suggests to formalize this property by using the TVD (total variation diminishing) criterion. With such criterion, the obtained schemes avoid oscillations. We can prove that the upwind scheme is TVD. Unfortunately, the TVD criterion is not applicable to systems.
In the following, we present a common way to build the numerical flux Φ_{ij}.
The value of the flux Φ_{ij} at the interface between two cells C_i and C_j can be determined precisely using a Godunov method which exactly solves the Riemann problem between the two cells. However, we do not use this method because of the expensive computation cost. We use instead approximate Riemann solvers. One of the best methods of approximate Riemann solvers is the Roe (1981) scheme which we will briefly describe below. This method consists in replacing the exact Riemann problem between two cells C_i and C_j by a locally linearized one.

1.4.2 Roe Scheme

Consider the Riemann problem at the interface between the two neighboring cells C_i and C_j respectively on the left and on the right of ∂C_{ij} :

$$\begin{cases} \frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{F}(\mathbf{U})) = 0 \\ \mathbf{U}(\mathbf{x}, t) = \begin{cases} \mathbf{U}_i & \text{if } \mathbf{x} \in C_i \\ \mathbf{U}_j & \text{if } \mathbf{x} \in C_j \end{cases} . \end{cases} \tag{1.14}$$

We rewrite the divergence term using three directions of space using the coordinates $(\mathbf{n}, \tau_1, \tau_2)$ with $\mathbf{n}$ exterior unit normal to ∂C_{ij} and τ_1, τ_2 the two unit tangents of that interface. We introduce the normal flux vector $\mathbf{F_n} = \mathbf{F}.\mathbf{n}$ and the tangent flux vectors $\mathbf{F}_{\tau_1} = \mathbf{F}.\tau_1$, $\mathbf{F}_{\tau_2} = \mathbf{F}.\tau_2$ which are the three columns of the flux matrix $\mathbf{F}$ written in coordinates $(\mathbf{n}, \tau_1, \tau_2)$. The equations of the system (1.14) can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F_n}}{\partial \mathbf{n}} + \frac{\partial \mathbf{F}_{\tau_1}}{\partial \tau_1} + \frac{\partial \mathbf{F}_{\tau_2}}{\partial \tau_2} = 0.$$

As the initial data is independent of τ_1 and τ_2, the solution of (1.14) will not depend on τ_1 and τ_2. We would like to determine an approximation of the normal flux crossing the interface using the one dimensional Riemann problem :

$$\begin{cases} \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F_n}}{\partial \mathbf{n}} = 0 \ \ \mathbf{U}(\mathbf{x}, t) = \begin{cases} \mathbf{U}_i & \text{if } \mathbf{x}.\mathbf{n} < 0 \\ \mathbf{U}_j & \text{if } \mathbf{x}.\mathbf{n} > 0 \end{cases} . \end{cases} \tag{1.15}$$

The term $\frac{\partial \mathbf{F_n}}{\partial \mathbf{n}}$ is linearized in the neighborhood of the interface between two cells C_i and C_j, and involves the Roe matrix $A(\mathbf{U}_i, \mathbf{U}_j, \mathbf{n}_{ij})$ which we denote by $A_{Roe,\mathbf{n}_{ij}}$. This matrix should satisfy the following conditions :

1. respect the hyperbolicity of the system, which means $A_{Roe,\mathbf{n}_{ij}}$ should be diagonalizable in $\mathbb{R}$;
2. be consistent with the jacobian matrix of the flux $\mathbf{F}$ in the direction $\mathbf{n}$:
$$A_{Roe,\mathbf{n}_{ij}}(\mathbf{U}_i, \mathbf{U}_i) = \nabla \mathbf{F}_{\mathbf{n}}(\mathbf{U}_i);$$
3. ensure the conservation principle across the boundary :
$$\forall (\mathbf{U}_i, \mathbf{U}_j),\ (\mathbf{F}(\mathbf{U}_i) - \mathbf{F}(\mathbf{U}_j)).\mathbf{n}_{ij} = A_{Roe,\mathbf{n}_{ij}}(\mathbf{U}_i - \mathbf{U}_j).$$

We remark that in general such a linearization is not uniquely defined.

The (first order accurate) numerical flux of Roe's can be given by :

$$\Phi_{ij} = \frac{\mathbf{F}(\mathbf{U}_i) + \mathbf{F}(\mathbf{U}_j)}{2}.\mathbf{n}_{ij} + |A_{Roe,\mathbf{n}_{ij}}|\frac{\mathbf{U}_i - \mathbf{U}_j}{2}, \tag{1.16}$$

where $|A_{Roe,\mathbf{n}_{ij}}|$ is the absolute value of the Roe matrix.
On a cartesian mesh, the Roe scheme is consistent with (1.14) and conservative. Hence the Lax-Wendroff theorem (given in LeVeque (2002), Section 12.10) implies that if the method converges, then it will converge to a weak solution of the conservation law (1.14).
The Roe scheme is a simple method which presents a minimal numerical diffusion, and which propagates all the waves with good speed. However, the computation of the absolute value of a matrix is not always obvious because we need to know the matrix and the sign of its eigenvalues. We will thereafter provide an efficient method for the computation of that value.

Let A be a diagonalizable matrix in $\mathbb{R}$, then we can write $A = TDT^{-1}$ with D the diagonal of A and we define $A^{\pm} = TD^{\pm}T^{-1}$ such that :

$$\text{diag}(D^+) = \{\lambda / \lambda = \lambda_k > 0\},$$

$$\text{diag}(D^-) = \{\lambda / \lambda = \lambda_k < 0\}.$$

We then have

$$A = A^+ + A^-,$$

and

$$A^+ = \frac{A + |A|}{2}, \tag{1.17}$$

$$A^- = \frac{A - |A|}{2}. \tag{1.18}$$

For the computation of A^+ et A^- we can use the following formulations :

$$A = \sum_{k=1}^{p} \lambda_k l_k \otimes r_k,$$

where λ_k are the eigenvalues of A and l_k, r_k are the corresponding left and right eigenvectors of A, respectively.
We then have the following expressions :

$$A^+ = \sum_{k=1}^{p} \lambda_k l_k \otimes r_k, \ \lambda_k > 0, \tag{1.19}$$

and

$$A^- = \sum_{k=1}^{p} \lambda_k l_k \otimes r_k, \ \lambda_k < 0. \tag{1.20}$$

By definition, $A_{Roe,\mathbf{n}_{ij}}$ is diagonalizable in $\mathbb{R}$ and we have :

$$A_{Roe,\mathbf{n}_{ij}} = T\text{diag}\,(\lambda_1, \ldots, \lambda_d)T^{-1},$$

then

$$\left|A_{Roe,\mathbf{n}_{ij}}\right| = T\text{diag}\,(|\lambda_1|, \ldots, |\lambda_d|)T^{-1}.$$

The formula (1.16) can be written as follows :

$$\begin{aligned}
\Phi_{ij} &= \mathbf{F}(\mathbf{U}_i)\mathbf{n}_{ij} + \frac{\mathbf{F}(\mathbf{U}_j) - \mathbf{F}(\mathbf{U}_i)}{2}\mathbf{n}_{ij} + \frac{1}{2}\left|A_{Roe,\mathbf{n}_{ij}}\right|(\mathbf{U}_i - \mathbf{U}_j) \\
&= \mathbf{F}(\mathbf{U}_i)\mathbf{n}_{ij} + \frac{1}{2}A_{Roe,\mathbf{n}_{ij}}(\mathbf{U}_j - \mathbf{U}_i) + \frac{1}{2}\left|A_{Roe,\mathbf{n}_{ij}}\right|(\mathbf{U}_i - \mathbf{U}_j) \\
&= \mathbf{F}(\mathbf{U}_i)\mathbf{n}_{ij} + A^-_{Roe,\mathbf{n}_{ij}}(\mathbf{U}_j - \mathbf{U}_i).
\end{aligned}$$

This flux can also be written in the following equivalent form :

$$\Phi_{ij} = \mathbf{F}(\mathbf{U}_j)\mathbf{n}_{ij} - A^+_{Roe,\mathbf{n}_{ij}}(\mathbf{U}_j - \mathbf{U}_i).$$

We inject those formulae and use the property $\sum \mathbf{F}(\mathbf{U}_i)\mathbf{n}_{ij} = \overrightarrow{0}$ to obtain our numerical scheme :

$$\frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i^n}{\Delta t} + \sum_{j\in N(i)} \frac{s_{ij}}{v_i} A^-_{Roe,\mathbf{n}_{ij}}(\mathbf{U}_j - \mathbf{U}_i) = 0. \tag{1.21}$$

The formulation (1.21) leads to two types of schemes : explicit and implicite schemes.

Explicit Scheme

$$\frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i^n}{\Delta t} + \sum_{j\in N(i)} \frac{s_{ij}}{v_i} A^-_{Roe,\mathbf{n}_{ij}}(\mathbf{U}^n)(\mathbf{U}_j^n - \mathbf{U}_i^n) = 0,$$

or

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \Delta t \sum_{j\in N(i)} \frac{s_{ij}}{v_i} A^-_{Roe,\mathbf{n}_{ij}}(\mathbf{U}^n)(\mathbf{U}_j^n - \mathbf{U}_i^n).$$

The advantage of the explicit scheme is clear. The solution at time step $(n+1)$ is explicitly computed using the solution at the previous time step

without matrix inversion. This method typically require less computational work and are simpler both in derivation, application and parallelization than implicit ones. Unfortunately, explicit schemes are generally characterized by a stability condition which is expressed by a CFL (Courant-Friedrichs-Lewy) condition. This condition depends on the fastest waves travelling the system and on the smallest cell of the mesh. In the context of high performance computing, the simulation of a large industrial problem often requires local refinements of the mesh to capture small scale phenomena such as boundary layers. Hence, the finer the local mesh is, the higher the number of time steps have to be performed using an explicit scheme.

Implicit Scheme

$$\frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i^n}{\Delta t} + \sum_{j \in N(i)} \frac{s_{ij}}{v_i} A^-_{Roe,\mathbf{n}_{ij}}(\mathbf{U}^{n+1})(\mathbf{U}_j^{n+1} - \mathbf{U}_i^{n+1}) = 0$$

Implicit methods, though computationally expensive, have less stringent stability conditions, and generally allow the use of larger time steps. Implicit schemes are usually chosen to capture solutions which require fine grid space, and to solve time scales that are not necessarily those of the fastest waves. For instance, in steady-state calculations we wish to integrate from some arbitrary initial guess to a steady solution which is often dominated by matter waves. Unfortunately, for implicit schemes, a non linear system has to be solved at each time step. We will therefore use Newton method to solve this system.

1.4.3 Newton Method for the Implicit Scheme

We denote by f the vector function with value in $\mathbb{R}^{N\times m}$ whose component i is

$$f(\mathbf{U})|_i = \frac{\mathbf{U}_i - \mathbf{U}_i^n}{\Delta t} + \sum_{j \in N(i)} \frac{s_{ij}}{v_i} \{A^-_{Roe}(\mathbf{U}))\}(\mathbf{U}_j - \mathbf{U}_i).$$

We have to find the zero of f assuming all $\mathbf{U}_i^n$ are known. For the sake of simplicity in the notations we remove the indices i. We start with $\mathbf{U}^0 = \mathbf{U}^n$, and we construct a sequence $(\mathbf{U}^k)_{k\geq 0}$ that tends to a zero of f. We compute $f(\mathbf{U}^{k+1})$ from $f(\mathbf{U}^k)$ by the Newton iterative algorithm :

$$f(\mathbf{U}^{k+1}) \cong f(\mathbf{U}^k) + f'(\mathbf{U}^k)(\mathbf{U}^{k+1} - \mathbf{U}^k) = 0.$$

We use an approximation of $f'(\mathbf{U}^k)$:

$$f'(\mathbf{U}^k) \simeq \Delta t^{-1}\mathbb{I} + \sum_i \sum_{j \in N(i)} \frac{s_{ij}}{v_i} \left(A^-(\mathbf{U}^k_{Roe})\right) e_{ij}$$

which means we do not differentiate the matrices A^- with regard to $\mathbf{U}$.
We have therefore the following system which results from Newton me-

thod :

$$\begin{aligned}\frac{\mathbf{U}_i^{k+1}-\mathbf{U}_i^k}{\Delta t} &+ \sum_j \frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k))\right]\left[(\mathbf{U}_j^{k+1}-\mathbf{U}_j^k)-(\mathbf{U}_i^{k+1}-\mathbf{U}_i^k))\right] \\ &+ \frac{U_i^k-\mathbf{U}_i^n}{\Delta t}+\sum_j \frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k)+\right](\mathbf{U}_j^k-\mathbf{U}_i^k)=0.\end{aligned}$$

By denoting $\delta\mathbf{U}^{k+1}=\mathbf{U}^{k+1}-\mathbf{U}^k$, the variation of the $k-th$ iteration that approximates the solution at time step $n+1$, we have a linear system of unknown $\delta\mathbf{U}^{k+1}$ to solve :

$$\begin{aligned}\frac{\delta\mathbf{U}^{k+1}}{\Delta t} &+ \sum_j \frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k)\right]\left(\delta\mathbf{U}_j^{k+1}-\delta\mathbf{U}_i^{k+1}\right) \\ &= -\frac{\mathbf{U}_i^k-\mathbf{U}_i^n}{\Delta t}-\sum_j \frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k)\right](\mathbf{U}_j^k-\mathbf{U}_i^k).\end{aligned}$$

Defining the unknown vector $\mathcal{U}=(\mathbf{U}_1,\ldots,\mathbf{U}_N)^t$, each Newton iteration for the computation of $\mathcal{U}$ at time step $n+1$ requires the numerical solution of the following linear system :

$$\mathcal{A}(\mathcal{U}^k)\delta\mathcal{U}^{k+1}=b(\mathcal{U}^n,\mathcal{U}^k). \tag{1.22}$$

1.4.4 Boundary Conditions

In order to make the problem well-posed (at least for the numerical simulation) we need to set the boundary conditions. We will see in this paragraph how to manage these conditions. The idea is to include a few additional cells on the boundary, called ghost cells, whose values are set at the beginning of each time step in some manner. Here, we are interested in four types of boundary conditions : solid wall, inlet, outlet, and Neumann boundary conditions.

Solid Wall Boundary Conditions In the case of solid wall boundary condition, the ghost cell is determined such that we impose a non friction at the boundary. So in our problem, if v_{in} is the velocity of the fluid in the boundary cell of the computational domain we define the velocity of the ghost cell $v_{out}.\mathbf{n}=-v_{in}.\mathbf{n}$ where $\mathbf{n}$ is the exterior normal vector and $v_{out}.\tau=v_{in}.\tau$ where τ is a tangent vector.

Neumann Boundary Conditions In this case, we prescribe the gradient normal to the boundary of a variable at the boundary. Usually, we take the state of the ghost cell equal to the internal state.

Inlet Boundary Conditions In this case, we prescribe the velocity and the temperature of the ghost cells. The pressure at the inlet is taken equal to that of the internal state.

Outlet Boundary Conditions At the outlet, the pressure is imposed while all other quantities are extrapolated from the interior domain.

1.4.5 Entropy Fix for Roe Scheme

It is well-known that the Roe's linearization may lead to non-entropic weak solutions of the governing equations due to the linearization of the Riemann problem. It often occurs when the exact solution consists a transonic rarefaction wave that is, when an eigenvalue λ_k of the jacobian matrix changes its sign (Toro (2009), Sec. 11.4). We are particularly interested in this situation because the eigenvalues of the jacobian matrix of cross flows (as we are interested in two-phase flows) may change sign (3.38).
In the Roe scheme, the numerical diffusion is proportional to the eigenvalues. Hence, when an eigenvalue is close to zero, the amount of numerical viscosity might be too small to prevent an entropy-condition violation. Several remedies are conceivable. Here we will outline the entropy fix presented by A.Harten (1983) .

Harten's Entropy Fix Harten's entropy fix (A.Harten (1983)) can be regarded as an addition of numerical diffusion to the kth field if the eigenvalue λ_k is too close to zero. Then instead of using $|A_{Roe,\mathbf{n}_{ij}}|$ in (1.16), we use $A_{H,\mathbf{n}_{ij}}$ with the same eigenvectors but the eigenvalue λ_k has been replaced by $\Phi_\delta(\lambda_k)$. $\Phi_\delta(\lambda_k)$ defined by :

$$\Phi_\delta(\lambda_k) = \begin{cases} |\lambda_k| & \text{if } |\lambda_k| \geq \delta \\ \frac{\lambda_k^2+\delta^2}{2\delta} & \text{if } |\lambda_k| < \delta \end{cases},$$

with δ a given positive constant. An advantage of this approach is its easy implementation as we use only the eigenvalues of the jacobian matrix. A disadvantage, on the other hand, is that the parameter δ must typically be depending on the problem.

Some Basics on Linear Systems and Their Solvers

2

Contents

In this chapter, we are interested in the solution of linear systems of dimension n

$$Ax = b, \tag{2.1}$$

where A is a nonsingular square matrix. Such systems can be encountered in almost every problem in scientific computing. Since the algorithms for solving linear systems are widely used in a large range of applications, the methods must then be efficient, accurate, reliable and robust.
Two classes of methods for solving those systems are of interest : direct methods and iterative methods. In a direct method, the matrix of the initial linear system is transformed or factorized into a simpler form, involving diagonal or triangular matrices, using elementary transformations, which can be solved easily. The exact solution is obtained in a finite number of arithmetic operations, if not considering numerical rounding errors. The most prominent direct method is the Gaussian elimination. Iterative methods, on the other hand, compute a sequence of approximate solutions, which converges to the exact solution in the limit, i.e., in practice until a desired accuracy is obtained.
For a long time, direct methods have been preferred to iterative methods for solving linear systems, mainly because of their simplicity and robustness. However, the emergence of conjugate gradient methods and Krylov subspace iterations provided an efficient alternative to direct solvers. Nowadays, iterative methods are almost mandatory in complex applications, notably because of memory and computational requirements that

prohibit the use of direct methods. Iterative methods usually involve a matrix-vector multiplication procedure that is cheap to compute on modern computer architectures.

2.1 Direct Solvers for Linear Systems

At first, we introduce direct methods to solve the linear system (2.1). We are especially interested in the LU factorization.

Definition 2.1 *To compute the* **factorization** *LU of the nonsingular matrix A is to determine L, a lower triangular matrix, and U, an upper triangular matrix, such that $A = LU$.*

Once the LU is obtained, the system $Ax = b$ can be solved in two steps : solving the system $Ly = b$, then the system $Ux = y$. Since L is a lower triangular matrix, the system $Ly = b$ can be solved by successive substitutions from the first to the last row. This step is called "forward substitution".
The forward substitution algorithm can be written as follows :

for $i = 1$ **to** n **do**
  for $j = 1$ **to** $i - 1$ **do**
    $b_i = b_i - l_{ij}.y_j$
  endfor
  if $l_{ii} = 0$ **then** *error !*
  else $y_i = b_i / l_{ii}$
endfor

Likewise, since U is an upper triangular matrix, the system $Ux = y$ can be solved by "backward substitution", which means successive substitutions from the last to the first row. The backward substitution algorithm can be written as follows :

for $i = n$ **to** 1 **do**
  for $j = i + 1$ **to** n **do**
    $y_i = y_i - u_{ij}.x_j$
  endfor
  if $u_{ii} = 0$ **then** *error !*
  else $x_i = y_i / u_{ii}$
endfor

The principal cost of direct methods lie on the factorization step. Here we give the algorithm of the Gauss factorization :

for $k = 1$ **to** $n - 1$ **do**
  for $i = k + 1$ **to** n **do**
    if $a_{kk} = 0$ **then** *error !*
    else $a_{ik} = a_{ik} / a_{kk}$
  endfor
  for $j = k + 1$ **to** n **do**
    for $i = k + 1$ **to** n **do**
      $a_{ij} = a_{ij} - a_{ik}.a_{kj}$
    endfor
  endfor
endfor

The algorithm builds, in the case of existence, the Gauss factorization of the matrix, $A = LU$, where L is a lower triangular matrix whose diago-

nal entries are 1 and U is an upper triangular matrix.

Unfortunately, not all matrices can be factorized into LU form. For example, it is easy to verify that $a_{11} = l_{11}u_{11}$. Therefore, if $a_{11} = 0$, then at least one of l_{11} and u_{11} has to be zero. The product LU becomes singular, while A is not necessarily singular. This means such matrices cannot be LU decomposed. To overcome this limitation, we use a LU decomposition with partial (or full) pivoting. That is, we left-multiply (or left and right multiply) A with a permutation matrix. This matrix reorders the rows (or rows and columns) of A. After reordering, the resulted matrix can be factorized in the LU form (Trefethen and D. Bau (1997), Lecture 20). As we often have to reorder the rows (or rows and columns) of the matrix A when performing the LU decomposition, it make the procedure very costly when we use parallel computing. In Okunev and Johnson (2005), the authors provided the necessary and sufficient conditions for existence of the LU factorization of an arbitrary matrix which states as follows :

Theorem 2.1 *A square matrix $A = (a_{ij})$ has an LU factorization iff it satisfies the conditions :*

$$rankA[\{1...k\}] + k \geq rankA[\{1...k\}, \{1...n\}] + rankA[\{1...n\}, \{1...k\}],$$

for all $k = 1, ..., n$.

2.2 Iterative Solvers for Linear Systems

If the size of the system (2.1) is very large, direct method are very expensive in memory and computational time. For example, the Gauss factorization algorithm has a complexity of $O(n^3)$ and there are no algorithms whose complexity is linear. Moreover, from a sparse matrix, when using direct methods we obtain a dense matrix which is not possible to be stored in memory.
In those cases, we use iterative methods. There are two main types of iterative methods : the classical and the projection ones.

2.2.1 Classical Iterative Methods

These methods are generally analyzed in terms of splitting the matrix A. Given a nonsingular matrix P, the splitting $A = P - N$ defines an iterative method as follows : Starting from an initial approximation x_0, we compute u_{m+1} by solving the following linear system :

$$Pu_{m+1} = Nx_m + b, \tag{2.2}$$

or, equivalently,

$$u_{m+1} = Bx_m + c, \tag{2.3}$$

where $B = P^{-1}N$ is the iteration matrix and $c = P^{-1}b$. P should be an "inexpensive" matrix. For example, in the Jacobi method, P is the diagonal part of A, and in the Gauss-Seidel method, P is the lower part of A ; see for instance [Varga (2000), chapters 3 and 4].
Since B does not depend on the iteration count m, these iterative methods are called *stationary*. The method (2.3) converges for any x_0 provided

$\rho(B) = \rho(\mathbb{I} - P^{-1}A) < 1$. Note that (2.2) can be rewritten in an equivalent form as

$$P(u_{m+1} - x_m) = r_m, \tag{2.4}$$

where $r_m \equiv b - Ax_m$ is the residual of the linear system at the m-th step. This leads to the class of Richardson methods ; see Varga (2000) and Saad (2000) for more details. Richardson methods contain all the main features of modern iterative methods : the scheme consists of matrix-vector products, vector-vector products, and vector updates, plus the solution of a linear system.
In modern times, classical iterative methods are seldom used in practice, whereas they are often employed as preconditioners for a more performing iterative procedure like the projection methods, covered in the following section.

2.2.2 Projection Methods

The idea of projection methods is to find an approximate solution from a well-chosen subspace of $\mathbb{R}^n$. This subspace, say $\mathcal{V}_m$, will be of dimension m (generally $m \ll n$) ; therefore the candidate solution will be found out by imposing m contraints. A typical way of describing these constraints is to impose m (indenpent) orthogonality conditions. In general, given an approximate solution x_m, the residual r_m is constrained to be orthogonal to m linearly indenpent vectors, which define another subspace, say $\mathcal{W}_m$, of dimension m. The space $\mathcal{W}_m$ is called the subspace of constraints.
Generally, we may define a projection method for the solution of (2.1) as a process which finds an approximate solution x_m by imposing the Petrov-Galerkin condition that x_m belongs to the affine space $x_0 + \mathcal{V}_m$ and that the new residual vector must be orthogonal to $\mathcal{W}_m$:

$$\begin{cases} \text{Find } x_m \in x_0 + \mathcal{V}_m \text{ such that} \\ b - Ax_m \perp \mathcal{W}_m. \end{cases} \tag{2.5}$$

To obtain a matrix representation of problem (2.5), let $V_m = [v_1, ..., v_m]$ be an nxm matrix whose column vectors form a basis of $\mathcal{V}_m$ and, similarly, $W_m = [w_1, ..., w_m]$ be an nxm matrix whose column vectors form a basis of $\mathcal{W}_m$. Writing the approximate solution as

$$x_m = x_0 + V_m y_m, \;\; y_m \in \mathbb{C}^m, \tag{2.6}$$

the orthogonality condition leads to the system of equations

$$W_m^T(b - Ax_0 - AV_m y_m = 0),$$

and therefore

$$x_m = x_0 + V_m(W_m^T A V_m)^{-1} W_m^T r_0.$$

Clearly, this approximate solution x_m is defined only when the matrix $W_m^T A V_m$ is nonsingular, which, for arbitrarily chosen spaces $\mathcal{V}_m$ and $\mathcal{W}_m$ is not guaranteed to be true even though A is nonsingular. It can be shown that $W_m^T A V_m$ is nonsingular for any basis V_m and W_m of $\mathcal{V}_m$ and $\mathcal{W}_m$, respectively, if A is positive definite and $\mathcal{W}_m \equiv \mathcal{V}_m$, or if A is nonsingular and $\mathcal{W}_m \equiv A\mathcal{V}_m$; see for instance, Saad (2000).

In modern iterative methods, the subspaces $\mathcal{V}_m$ and $\mathcal{W}_m$ are Krylov subspaces, that is subspaces spanned by vectors of the form $\mathbb{P}_m(A)v$, where $\mathbb{P}_m(A)$ is a polynomial of degree m in A, and v is a vector. The Krylov subspace $\mathcal{V}_m$ may be defined as

$$\mathcal{K}_m = \mathcal{K}_m(A; r_0) = \text{span}\{r_0, Ar_0, ..., A^{m-1}r_0\}.$$

The main classes of Krylov subspace iterative methods fall into three categories, depending on the choice of $\mathcal{W}_m$:

- *The Ritz-Galerkin approach* ($\mathcal{W}_m \equiv \mathcal{V}_m$) : build x_m for which the residual is orthogonal to the current space. When the matrix A is symmetric positive definite, this choice minimises the A norm of the error $x_m - u$. An example is the Conjugate Gradient (CG) method of Hestenes and Steifel (1952) ;

- *The minimum residual approach* ($\mathcal{W}_m \equiv A\mathcal{V}_m$) : determine x_m which minimises the residual norm $||b - Ax_m||_2$. Examples are the GMRES method of Saad and Schultz and its variants (Saad (2000)) ;

- *The Petrov-Galerkin approach* : find x_m so that the residual is orthogonal to some other m-dimensional space. A possible choice is $\mathcal{W}_m \equiv \mathcal{K}_m(A^T, \tilde{r}_0)$, where $\tilde{r}_0$ is a vector nonproportional to r_0. This choice was designed for nonsymmetric problems, and may provide a short-term recurrence relation for the Krylov subspace bases. Examples are the Bi-CG method of Fletcher (1976), the CGS method of Sonneveld (1989), the Bi-CGSTAB method of van der Vorst (1992), and the QMR method of Freund and Nachtigal (1991).

As a general rule, we can say that if the matrix is symmetric positive definite, the CG method is the best choice, while for nonsymmetric problems usually GMRES is the most stable of all methods. Many other methods have been proposed in literature, like FOM, ORTHOMIN, ORTHODIR, ORTHORES, MINRES, SYLMMLQ. For an exhaustive presentation the reader is addressed to Saad (2000).

GMRES Method In the sequel, we describe in some details the GMRES method. This method was proposed by Saad and Schultz (1986) for the solution of large nonsymmetric problems.
For the sake of generality, we describe this method for linear systems whose entries are complex. Everything also applies to real entries.
Let $x_0 \in \mathbb{C}^n$ be an initial guess for the linear system (2.1) and $r_0 = b - Ax_0$ be its corresponding residual. At step m, the GMRES algorithm builds an approximation of the solution of (2.1) under the form

$$x_m = x_0 + V_m y_m, \tag{2.7}$$

where $y_m \in \mathbb{C}^m$ and $V_m = [v_1, ..., v_m]$ is an orthonormal basis for the Krylov space of dimension m defined by

$$\mathcal{K}_m = \mathcal{K}_m(A; r_0) = \text{span}\{r_0, Ar_0, ..., A^{m-1}r_0\}.$$

The vector y_m is determined so that the 2-norm of the residual $r_k = b - Ax_k$ is minimized over $x_0 + \mathcal{K}_m$. The basis V_m for the Krylov subspace

$\mathcal{K}_m$ is obtained via the well-known Arnoldi process (Arnoldi (1951)). The orthogonal projection of A onto $\mathcal{K}_m$ results in an upper Hessenberg matrix $H_m = V_m^T A V_m$ of order m. The Arnoldi process satisfies the relationship

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_k^T, \tag{2.8}$$

where e_m is the m^{th} canonical basis vector. Equation (2.8) can be rewritten in a matrix form as

$$AV_m = V_{m+1} \bar{H}_m,$$

where

$$\bar{H}_m = \begin{bmatrix} H_m \\ 0...0 \; h_{m+1,m} \end{bmatrix}$$

is an $(m+1) \times m$ matrix.
Let $v_1 = \frac{r_0}{\beta}$ where $\beta = ||r_0||_2$. The residual r_m which is associated with the approximate solution x_m defined by (2.7), satisfies

$$\begin{aligned} r_m &= b - Ax_m = b - A(x_0 + V_m y_m) \\ &= r_0 - AV_m y_m = r_0 - V_{m+1} \bar{H}_m y_m \\ &= V_{m+1}(\beta e_1 - \bar{H}_m y_m). \end{aligned}$$

Because V_{m+1} is a matrix with orthonormal columns, the residual norm $||r_m||_2 = ||\beta e_1 - \bar{H}_m y_m||_2$ is minimized when y_m solves the linear least-squares problem

$$\min_{y \in \mathcal{C}^m} ||\beta e_1 - \bar{H}_m y||_2. \tag{2.9}$$

We denote by y_m the solution of (2.9). Therefore, $x_m = x_0 + V_m y_m$ is an approximate solution of (2.1) for which the residual is minimized over $x_0 + \mathcal{K}_m$. The GMRES method owes its name to this minimization property that is its key feature as it ensures the decrease of the residual norm associated with the sequence of iterates.

An appealing property of GMRES is that, in exact arithmetic, the method cannot breakdown or, more precisely, it can only breakdown when it delivers the exact solution. However, GMRES is not limited-memory : V_m has to be stored completely. Every iteration, a new basis vector has to be computed and stored. Orthogonalization of new basis vectors also becomes increasingly more expensive with m. A restarted GMRES can be employed to reduce memory requirements, at the price of possibly non-convergence of the scheme. This restarted method is usually called GMRES(k), where k is the maximal dimension of the Krylov subspace.

For the convergence of GMRES method, from Trefethen and D. Bau (1997), we have

$$\frac{\|r_m\|}{\|b\|} \leq \inf_{p \in P_m} \|p_m(A)\| \leq \kappa(V) \inf_{p \in P_m} \max_{\lambda \in \sigma(A)} |p(\lambda)|, \tag{2.10}$$

where P_m denotes the set of polynomials of degree at most m with $p(0) = 1$, V is a nonsingular matrix of eigenvectors of A (assuming A is diagonalizable), and $\sigma(A)$ is the set of eigenvalues of A. Roughly speaking, this says that fast convergence occurs when the eigenvalues of A are clustered away from the origin and A is not too far from normality.

2.3 Preconditioner

The convergence of iterative methods depends on the spectral properties of the linear system matrix. We define the condition number of the matrix by $K(A) = \|A\|\|A^{-1}\|$. The basic idea of preconditioning is to replace the original problem (2.1) by

$$M^{-1}Ax = M^{-1}b$$

(left-preconditioning), or by

$$AM^{-1}Mx = b$$

(right-preconditioning), using a linear transformation M^{-1}, called preconditioner, in order to reduce the condition number of the preconditioned matrix. In general terms, a preconditioner is any kind of transformation applied to the original system which makes it easier to solve.
In a modern prespective, the general problem of finding an efficient preconditioner is to identify a linear operator M having the following properties :

1. M is a good approximation of A in some sense. Although no general theory is available, we can say the M should act so that $M^{-1}A$ is near to being the identity matrix and its engenvalues are clustered within a sufficiently small region of the complex plane ;

2. M is efficient, in the sense that the iteration method converges much faster, in terms of CPU time, for the preconditioned system. In other words, preconditioners must be selected in such a way that the cost of constructing and using them is offset by the improved convergence properties they allow to achieve ;

3. M or M^{-1} can take advantage of the architecture of modern supercomputers, that is, can be constructed and applied in parallel environments.

The choice of M varies from "black-box" algebraic techniques which can be applied to general matrices to "problem dependent" preconditioners which exploit special features of a particular class of problems.

2.3.1 Incomplete LU Preconditioners

A broad class of effective preconditioners for sparse matrices is based on incomplete factorization of the matrix, and it is usually indicated as ILU. A general algorithm for building ILU preconditioner can be derived by performing Gaussian elimination and dropping some elements in predetermined nondiagonal positions. This algorithm can be written as follows :
for $k = 1$ **to** $n - 1$ **do** *:*
  for $i = k + 1$ **to** n **do**
    if $a_{ik} \neq 0$ *:*
    if $a_{kk} = 0$ **then** *error !*

else $a_{ik} = a_{ik}/a_{kk}$
for $j = k+1$ **to** n **do** *:*
if $a_{ij} \neq 0$ **then** $a_{ij} = a_{ij} - a_{ik}a_{kj}$
endfor
endfor
endfor

Here, the matrix which we implicitly define preserve the sparse structure of A. This simple ILU factorization is known as ILU(0). Although effective, in some cases the accuracy of the ILU(0) may be insufficient to yield an adequate rate of convergence. More accurate factorizations will differ from ILU(0) by allowing some fill-in. The resulting class of methods is called ILU(k) where k is the level of fill-in. A level of fill-in is attributed to each matrix entry that occurs in the incomplete factorization process. Fill-ins are dropped based on the value of the level of fill. The level of fill should be indicative of the size of the element : the higher the level of fill, the less sparse the matrix. This method is detailed in Saad (2000).

Alternative dropping techniques can be based on the numerical size of the element to be discarded. The general strategy is to compute an entire row of the L and U matrices, and then keep only the entries that larger than a threshold. In this way, the amount of fill-in is controlled ; however, the structure of the resulting matrices is undefined. That factorization is referred to as ILUT. In the following algorithm, w is a full-length working row which is used to accumulate linear combinations of sparse rows in the elimination, w_k is the k^{th} entry of this row and a_{i*} denotes the i^{th} row of A.

for $i = 1$ **to** n **do** *:*
$w := a_{i*}$
for $k = 1$ **to** $i-1$ **do**
if $w_k \neq 0$ *:*
if $a_{kk} = 0$ **then** *error !*
else $w_k = w_k/a_{kk}$
Apply a dropping rule to w_k
if $w_k \neq 0$ *:* **then** $w = w - w_k u_{k*}$
endfor
Apply a dropping rule to w
for $j = 1$ **to** $i-1$ **do** *:*
$l_{ij} = w_j$
endfor
for $j = i$ **to** n **do** *:*
$u_{ij} = w_j$
endfor
$w := 0$
endfor

Notwishstanding their popularity, incomplete factorization methods have their limitations. In fact, one can only prove the existence of the ILU preconditioner for an M-matrix (Meijerink and van der Vorst (1977)) and a H-matrix (Varga et al. (1980)). Also, the ILU factorization may breakdown or become indefinite for a positive matrix. They also add more difficulties in the parallelization, and lack of algorithmic scalability.

Many other variants have been presented in the literature. Latest developments concern multilevel methods, like ILUM (Saad (2000)), or BILUM (Saad and Zhang (1999)), or ARMS (Saad and Suchomel (2002)). These approaches try to combine the "general-purpose" approach of ILU-type preconditioners with the superior convergence rate of multilevel methods, and of multigrid methods, using some ideas from domain decomposition methods.

2.3.2 Schur Complement Techniques

Let I and B be two disjoint sets of indices, whose cardinality is n_I and n_B, respectively, such that a generic vector $x \in \mathbb{R}^n$ can be written as $x = [x_I, x_B]$, and a generic matrix $A \in \mathbb{R}^{nxn}$ can be reordered consistently with I and B,

$$A = \begin{pmatrix} A_{II} & A_{IB} \\ A_{BI} & A_{BB} \end{pmatrix} \tag{2.11}$$

where A_{II} and A_{BB} are square matrices.

Definition 2.2 *If A_{II} is nonsingular, we define*

$$S \equiv A/A_{II} = A_{BB} - A_{BI}A_{II}^{-1}A_{IB}. \tag{2.12}$$

If A_{II} is singular, we define

$$S \equiv A/A_{II} = A_{BB} - A_{BI}A_{II}^{+}A_{IB}, \tag{2.13}$$

where A_{II}^{+} is the pseudo-inverse (Moore-Penrose) inverse of matrix A_{II}. S is called the Schur complement of A with respect to A_{II}.

Note that S is the matrix that results when we eliminate the block A_{BI} using block Gaussian elimination with A_{II} as pivot block. In fact, the block matrix triangular factorisation of A is readily found to be

$$A = \begin{pmatrix} \mathbb{I}_I & 0 \\ A_{BI}A_{II}^{-1} & \mathbb{I}_B \end{pmatrix} \begin{pmatrix} A_{II} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} \mathbb{I}_I & A_{II}^{-1}A_{IB} \\ 0 & \mathbb{I}_B \end{pmatrix}, \tag{2.14}$$

where $\mathbb{I}_I$ and $\mathbb{I}_B$ are the identity matrices whose dimensions are n_I and n_B respectively. Then, the inverse of A may be expressed using A_{II}^{-1} and the inverse of Schur's complement (if it exists) as

$$A^{-1} = \begin{pmatrix} \mathbb{I}_I & -A_{II}^{-1}A_{IB} \\ 0 & \mathbb{I}_B \end{pmatrix} \begin{pmatrix} A_{II}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} \mathbb{I}_I & 0 \\ -A_{BI}A_{II}^{-1} & \mathbb{I}_B \end{pmatrix} \tag{2.15}$$

or, equivalently

$$A^{-1} = \begin{pmatrix} A_{II}^{-1} + A_{II}^{-1}A_{IB}S^{-1}A_{BI}A_{II}^{-1} & -A_{II}^{-1}A_{IB}S^{-1} \\ -S^{-1}A_{BI}A_{II}^{-1} & S^{-1} \end{pmatrix}. \tag{2.16}$$

Then if we can approximate the inverse of A_{II} and S we can use (2.16) to approximate the inverse of A and use that as a preconditioner.

Numerical Simulation of Compressible flows

3

Contents

In this chapter, we first describe in Sec. 3.1, the mathematical and computational framework of the compressible single-phase flows considered for our purpose. Using the finite volume method described in Sec. 1.4, we obtain conservative schemes for the compresible single-phase flow. We then present different test cases utilized from now on. Those tests are academic ones and allow us to verify our schemes and study the spectrum of our matrices. The Scaling strategy, which improves the condition number of the matrix, is introduced in Sec. 3.1.3. This strategy is a similarity transformation of the matrix, so that the off diagonal entries of the matrix have the same order of magnitude. Numerical results show that this method reduces the computational cost and accelerates the convergence of both linear system and Newton scheme iterations.

The mathematical and computational framework of the compressible two-phase flows are presented in Sec.3.2. As in the case of single-phase flows, we separate the convection flux and the diffusion flux. We pay special attention to the discretization of the convective flux. Since the convective flux consists of nonconservative terms, we need to find an

appropriate way to locally linearize those terms. We also study the asymptotic behavior of the isentropic two-fluid model in the case where the two phases are assumed incompressible. That hypothesis gives a system of two partial differential equations of first order. This system allows us to easily study its hyperbolicity and its behavior when one of the phases vanishes. Also, we introduce an entropy fix that helps obtaining a positive Roe's scheme. We then present the numerical test cases and numerical resutls.

3.1 Single-phase Flows

3.1.1 Compressible Navier-Stoke equations

In mechanics, the Navier-Stokes equations describe the motion of fluid substances. These equations arise from applying Newton's second law to fluid motion, together with the assumption that the fluid stress is the sum of a diffusive viscous term and a pressure term. There exists various forms of the Navier-Stokes equations. For the sake of simplicity, we present here a simple form. It consists of the following three balance laws for mass, momentum and energy :

$$\left\{\begin{array}{rcrcrcl} \frac{\partial \rho}{\partial t} & + & \nabla.\mathbf{q} & & & = & 0 \\ \frac{\partial \mathbf{q}}{\partial t} & + & \nabla.\left(\mathbf{q}\otimes\frac{\mathbf{q}}{\rho}+p\mathbb{I}_d\right) & - & \nu\Delta(\frac{\mathbf{q}}{\rho}) & = & 0 \\ \frac{\partial(\rho E)}{\partial t} & + & \nabla.\left[(\rho E+p)\frac{\mathbf{q}}{\rho}\right] & - & \lambda\Delta T & = & 0, \end{array}\right. \tag{3.1}$$

where ρ is the density, $\mathbf{v}$ is the velocity vector, $\mathbf{q}=\rho\mathbf{v}$ is the momentum, p is the pressure, ρe is the internal energy, $\rho E=\rho e+\frac{||\mathbf{q}||^2}{2\rho}$ is the total energy, T is the absolute temperature, ν is the viscosity, and λ is the thermal conductivity.

In order to close the systeme, we have to add an equation of state (EOS) which is a relation between the density, the velocity and the pressure of the fluid. For ideal gases with the ratio of specific heats γ ($\gamma>0$), this relation is written as :

$$p=(\gamma-1)\rho e. \tag{3.2}$$

By denoting $\mathbf{U}=\begin{pmatrix}\rho\\ \mathbf{q}\\ E\end{pmatrix}$, the Navier-Stokes equations can be written as a system of conservation laws :

$$\frac{\partial \mathbf{U}}{\partial t}+\nabla\cdot(\mathbf{F}^{conv}(\mathbf{U}))+\nabla\cdot\left(\mathbf{F}^{diff}(\mathbf{U})\right)=0, \tag{3.3}$$

where

$$\mathbf{F}^{conv}(\mathbf{U})=\begin{pmatrix}\mathbf{q}\\ \mathbf{q}\otimes\frac{\mathbf{q}}{\rho}+p\mathbb{I}_d\\ (\rho E+p)\frac{\mathbf{q}}{\rho}\end{pmatrix}$$

is the convective flux, and

$$\mathbf{F}^{diff}(\mathbf{U})=\begin{pmatrix}0\\ -\nu\overrightarrow{\nabla}(\frac{\mathbf{q}}{\rho})\\ -\lambda\overrightarrow{\nabla}T\end{pmatrix}$$

is the diffusive flux.
We would like to set the (3.3) in the quasilinear form (3.4) using an orthonormal basis $(\mathbf{n},\tau_1,\cdots\tau_{d-1})$ of the d dimensional space $\mathbb{R}^d$:

$$\begin{aligned}\frac{\partial \mathbf{U}}{\partial t}+A_{\mathbf{n}}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial \mathbf{n}}+A_{\tau_1}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial \tau_1}+\cdots A_{\tau_{d-1}}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial \tau_{d-1}}\\ +D_{\mathbf{n}}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial \mathbf{n}}+D_{\tau_1}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial \tau_1}+\cdots D_{\tau_{d-1}}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial \tau_{d-1}}=0.\end{aligned} \tag{3.4}$$

As we use the finite volume method, we will focus on the determination of $A_{\mathbf{n}}$ and $D_{\mathbf{n}}$ with $\mathbf{n}$ is the unit vector normal to the interface between two neighboring cells.
Because we are only interested in $A_{\mathbf{n}}$ and $D_{\mathbf{n}}$, we will only regard the terms involving $\frac{\partial \mathbf{U}}{\partial \mathbf{n}}$, and for the simplicity of writing, we often replace contributions of $\frac{\partial \mathbf{U}}{\partial \tau_i}$ by $\cdots$. As an example of this convention, the equation 3.4 can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + A_{\mathbf{n}}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial \mathbf{n}} + \cdots + D_{\mathbf{n}}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial \mathbf{n}} + \cdots = 0,$$

where the coefficient matrices $A_{\mathbf{n}}$ and $D_{\mathbf{n}}$ are the Jacobian matrices

$$A_{\mathbf{n}} = \frac{\partial \mathbf{F}^{conv}(\mathbf{U})}{\partial \mathbf{n}}, \quad D_{\mathbf{n}} = \frac{\partial \mathbf{F}^{diff}(\mathbf{U})}{\partial \mathbf{n}}.$$

Analysis of the convective part The Jacobian matrix $A_{\mathbf{n}}$ is

$$A_{\mathbf{n}}(\mathbf{U}) = \begin{pmatrix} 0 & \mathbf{n}^t & 0 \\ \frac{\gamma-1}{2}\frac{\|\mathbf{q}\|^2}{\rho^2}\mathbf{n} - \mathbf{v}\cdot\mathbf{n}\frac{\mathbf{q}}{\rho} & \frac{\mathbf{q}}{\rho}\otimes\mathbf{n} - (\gamma-1)\mathbf{n}\otimes\frac{\mathbf{q}}{\rho} + \mathbf{n}.\frac{\mathbf{q}}{\rho}\mathbb{I}_d & (\gamma-1)\mathbf{n} \\ (\frac{\gamma-1}{2}\frac{\|\mathbf{q}\|^2}{\rho^2} - H)\mathbf{v}\cdot\mathbf{n} & H\mathbf{n}^t - (\gamma-1)\mathbf{v}\cdot\mathbf{n}\frac{\mathbf{q}}{\rho}^t & \gamma\mathbf{v}\cdot\mathbf{n} \end{pmatrix}, \tag{3.5}$$

where H is the total enthalpy and is defined by

$$H = \gamma E + \frac{\gamma-1}{2}\frac{||\mathbf{q}||^2}{\rho^2}.$$

The eigenvalues λ_k of that matrix are given as follows :

$$\begin{cases} \lambda_1 = \mathbf{v}\cdot\mathbf{n} - c \\ \lambda_2 = \cdots = \lambda_{d+1} = \mathbf{v}\cdot\mathbf{n} \\ \lambda_{d+2} = \mathbf{v}\cdot\mathbf{n} + c, \end{cases}$$

where c is the sound speed.
The left eigenvectors l_k and right eigenvectors r_k associated to these eigenvalues can be taken equal to

– $\mathbf{v}\cdot\mathbf{n} - c$:

$$r_1 = \begin{pmatrix} 1 \\ \frac{\mathbf{q}}{\rho} - c\,\mathbf{n} \\ H - \mathbf{v}\cdot\mathbf{n}c \end{pmatrix} \text{ and } l_1 = \frac{1}{2c^2}\begin{pmatrix} \frac{\gamma-1}{2}\frac{\|\mathbf{q}\|^2}{\rho^2} + \mathbf{v}\cdot\mathbf{n}c \\ (1-\gamma)\frac{\mathbf{q}}{\rho} - c\,\mathbf{n}_{ij} \\ (\gamma-1) \end{pmatrix},$$

– $\mathbf{v}\cdot\mathbf{n} + c$:

$$r_{d+2} = \begin{pmatrix} 1 \\ \frac{\mathbf{q}}{\rho} + c\,\mathbf{n} \\ H + \mathbf{v}\cdot\mathbf{n}\,c \end{pmatrix} \text{ and } l_{d+2} = \frac{1}{2c^2}\begin{pmatrix} \frac{\gamma-1}{2}\frac{\|\mathbf{q}\|^2}{\rho^2} - \mathbf{v}\cdot\mathbf{n}c \\ (1-\gamma)\frac{\mathbf{q}}{\rho} + c\,\mathbf{n} \\ (\gamma-1) \end{pmatrix},$$

– $\mathbf{v}\cdot\mathbf{n}$:

$$r_2 = \begin{pmatrix} 1 \\ \frac{\mathbf{q}}{\rho} \\ H - \frac{c^2}{\gamma-1} \end{pmatrix} \text{ and } l_2 = \frac{\gamma-1}{c^2}\begin{pmatrix} H - \frac{\|\mathbf{q}\|^2}{\rho^2} \\ \frac{\mathbf{q}}{\rho} \\ -1 \end{pmatrix},$$

then, let $(e_p)_{1\leq p\leq d-1}$ be an orthonormal basis of the hyperplane orthogonal to $\mathbf{n}$. We have :

$$r_{p+2} = \begin{pmatrix} 0 \\ e_p \\ \frac{\mathbf{q}}{\rho}.e_p \end{pmatrix} \text{ and } l_{p+2} = \begin{pmatrix} -\frac{\mathbf{q}}{\rho}\cdot e_p \\ e_p \\ 0 \end{pmatrix}.$$

As the eigenvalues λ_k are all real and the eigenvectors r_k form a complete set of linearly independent eigenvectors, the convective part of the Navier-Stokes equations (3.1) is hyperbolic.

Analysis of the diffusive part The Jacobian matrix $D_{\mathbf{n}}$ of the diffusive part is

$$D_{\mathbf{n}}(\mathbf{U}) = \begin{pmatrix} 0 & \mathbf{0} & 0 \\ \frac{\nu\mathbf{q}}{\rho^2} & \frac{-\nu}{\rho}\mathbb{I}_d & 0 \\ \frac{\lambda}{C_v}\left(\frac{C_v T}{\rho} - \frac{||\mathbf{q}||^2}{2\rho^3}\right) & \frac{\mathbf{q}\,{}^t\lambda}{\rho^2 C_v} & -\frac{\lambda}{C_v\rho} \end{pmatrix},$$

where C_v is the heat capacity at constant volume.

3.1.2 Numerical Scheme

The conservation form (3.3) allows us to use the numerical scheme described in Sec.1.4. We decompose the computational domain into N disjoint cells C_i with volume v_i. Integrating the system (3.3) over C_i,

$$\int_{C_i}\left\{\frac{\partial \mathbf{U}}{\partial t} + \nabla\cdot\left(\mathbf{F}^{conv}(\mathbf{U})\right) + \nabla\cdot\left(\mathbf{F}^{diff}(\mathbf{U})\right)\right\}\mathrm{d}\mathbf{x} = 0$$

yields

$$\int_{C_i}\frac{\partial \mathbf{U}}{\partial t}\,\mathrm{d}\mathbf{x} + \sum_{j\in N(i)}\int_{\partial C_{ij}}\mathbf{F}^{conv}(\mathbf{U}).\mathbf{n}_{ij}\,\mathrm{d}s + \sum_{j\in N(i)}\int_{\partial C_{ij}}\mathbf{F}^{diff}(\mathbf{U}).\mathbf{n}_{ij}\,\mathrm{d}s = 0, \tag{3.6}$$

or, equivalently

$$\int_{C_i}\frac{\partial \mathbf{U}}{\partial t}\,\mathrm{d}\mathbf{x} + \sum_{j\in N(i)} s_{ij}\Phi^{conv}_{ij} + \sum_{j\in N(i)} s_{ij}\Phi^{diff}_{ij} = 0, \tag{3.7}$$

where ∂C_{ij} is the common boundary of two neighboring cells C_i and C_j, $N(i)$ is the set of neighbors of a given cell C_i and $\mathbf{n}_{ij}$ the exterior unit normal vector of ∂C_{ij}. We also denote s_{ij} the area of ∂C_{ij}.
We will build the numerical fluxes of convection and diffusion separately.

Convection Discretization We define the numerical flux of convection

$$\Phi^{conv}_{ij} = \frac{1}{s_{ij}}\int_{\partial C_{ij}}\left(\mathbf{F}^{conv}(\mathbf{U})\right).\mathbf{n}_{ij}\mathrm{d}s,$$

using the Roe flux as follows :

$$\begin{aligned}
\Phi^{conv}_{ij} &= \frac{\mathbf{F}^{conv}(\mathbf{U}_i) + \mathbf{F}^{conv}(\mathbf{U}_j)}{2}.\mathbf{n}_{ij} + \mathcal{D}_{ij}\frac{\mathbf{U}_i - \mathbf{U}_j}{2} && (3.8)\\
&= \mathbf{F}^{conv}(\mathbf{U}_i).\mathbf{n}_{ij} + A^{-}_{ij}(\mathbf{U}_j - \mathbf{U}_i) && (3.9)\\
&= \mathbf{F}^{conv}(\mathbf{U}_j).\mathbf{n}_{ij} - A^{+}_{ij}(\mathbf{U}_j - \mathbf{U}_i), && (3.10)
\end{aligned}$$

where $\mathcal{D}_{ij}$ is an upwinding matrix, $A_{Roe,\mathbf{n}_{ij}}$ is the Roe matrix and $A^{\pm}_{ij} = \frac{1}{2}(A_{Roe,\mathbf{n}_{ij}} \pm \mathcal{D}_{ij})$. The choice $\mathcal{D}_{ij} = 0$ gives the centered scheme, whereas $\mathcal{D}_{ij} = |A_{Roe,\mathbf{n}_{ij}}|$ gives the upwind scheme.

We now have to determine the Roe matrix $A^{-}_{Roe,\mathbf{n}_{ij}}$. In order to determine the Roe matrix we use the following Roe-averaged state :

Definition 3.1 *The state $\mathbf{U}_{Roe,(i,j)}$ is called the Roe-averaged state of two states $\mathbf{U}_i$ and $\mathbf{U}_j$ provided that*

$$A_{Roe,\mathbf{n}_{ij}} = A(\mathbf{U}_{Roe,(i,j)}, \mathbf{n}_{ij}),$$

where $A(\mathbf{U}, \mathbf{n})$ is the Jacobian matrix of $\mathbf{F}^{conv}(\mathbf{U}).\mathbf{n}$.

If the states $\mathbf{U}_i$ and $\mathbf{U}_j$ have densities, velocities and enthapies $\rho_i,\ \mathbf{v}_i\ H_i$ and $\rho_j,\ \mathbf{v}_j\ H_j$ respectively , the Roe-averaged state $\mathbf{U}_{Roe,(i,j)}$ is given by :

$$\rho_{Roe,(i,j)} = \sqrt{\rho_i \rho_j}, \tag{3.11}$$

$$\mathbf{v}_{Roe,(i,j)} = \frac{\sqrt{\rho_i}\mathbf{v}_i + \sqrt{\rho_j}\mathbf{v}_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}, \tag{3.12}$$

$$H_{Roe,(i,j)} = \frac{\sqrt{\rho_i}H_i + \sqrt{\rho_j}H_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}. \tag{3.13}$$

For the computation of the Roe-averaged state, one can see (Godlewski and Raviart (1996), page 211).
We now study the matrix $A(\mathbf{U}_{Roe,(i,j)}, \mathbf{n}_{ij})$.
We can build A_{Roe} explicitly :

$$A_{Roe} = \begin{pmatrix} 0 & \mathbf{n}_{ij}^t & 0 \\ \frac{\gamma-1}{2}\frac{\|\mathbf{q}\|^2}{\rho^2}\mathbf{n}_{ij} - \mathbf{v}\cdot\mathbf{n}_{ij}\frac{\mathbf{q}}{\rho} & \frac{\mathbf{q}}{\rho}\otimes\mathbf{n}_{ij} - (\gamma-1)\mathbf{n}_{ij}\otimes\frac{\mathbf{q}}{\rho} + \mathbf{n}_{ij}.\frac{\mathbf{q}}{\rho}\mathbb{I}_d & (\gamma-1)\mathbf{n}_{ij} \\ (\frac{\gamma-1}{2}\frac{\|\mathbf{q}\|^2}{\rho^2} - H)\mathbf{v}\cdot\mathbf{n}_{ij} & H\mathbf{n}_{ij}^t - (\gamma-1)\mathbf{v}\cdot\mathbf{n}_{ij}\frac{\mathbf{q}}{\rho}^t & \gamma\mathbf{v}\cdot\mathbf{n}_{ij} \end{pmatrix}.$$

As in the equations (1.19) and (1.20) of chapter 1, we have :

$$A_{Roe} = \sum_{k=1}^{n} \lambda_k l_k \otimes r_k,$$

$$A^{-}_{Roe} = \sum_{k=1}^{n} \lambda_k l_k \otimes r_k,\ \lambda_k < 0.$$

Diffusion Discretization In a similar way, we can define the numerical flux of diffusion :

$$\Phi^{diff}_{ij} = \frac{1}{s_{ij}} \int_{\partial C_{ij}} \mathbf{F}^{diff}(\mathbf{U}).\mathbf{n}_{ij} ds, \tag{3.14}$$

and choose the following discretization that is exact on cartesian meshes

$$\Phi^{diff}_{ij} = D(\mathbf{U}_{diff,(i,j)})(\mathbf{U}_j - \mathbf{U}_i). \tag{3.15}$$

In the following, we introduce the interface state for diffusion $\mathbf{U}_{diff,(i,j)}$ and discretize the differential operators at the interface to obtain the diffusion matrix D.

Definition 3.2 *If* $\mathbf{U}_i = \begin{pmatrix} \rho_i \\ \mathbf{q}_i \\ \rho_i E_i \end{pmatrix}$ *and* $\mathbf{U}_j = \begin{pmatrix} \rho_j \\ \mathbf{q}_j \\ \rho_j E_j \end{pmatrix}$, *the state*

$$\mathbf{U}_{diff,(i,j)} = \frac{1}{2} \begin{pmatrix} \rho_i + \rho_j \\ \mathbf{q}_i + \mathbf{q}_j \\ \rho_i E_i + \rho_j E_j \end{pmatrix}$$

is called the interface diffusion state of $\mathbf{U}_i$ *and* $\mathbf{U}_j$.

We then obtain the diffusion matrix

$$D(\mathbf{U}) = \begin{pmatrix} 0 & \mathbf{0} & 0 \\ \frac{\nu \mathbf{q}}{\rho^2} & \frac{-\nu}{\rho}\mathbb{I}_d & 0 \\ \frac{\lambda}{C_v}\left(\frac{C_v T}{\rho} - \frac{\|\mathbf{q}\|^2}{2\rho^3}\right) & \frac{\mathbf{q}^t \lambda}{\rho^2 C_v} & -\frac{\lambda}{C_v \rho} \end{pmatrix}.$$

Using the numerical scheme (3.7), we obtain two types of schemes :

Explicit scheme

$$\frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i^n}{\Delta t} + \sum_{j \in N(i)} \frac{s_{ij}}{v_i} \{A^-(\mathbf{U}^n_{Roe,(i,j)}) + D(\mathbf{U}^n_{diff,(i,j)})\}(\mathbf{U}_j^n - \mathbf{U}_i^n) = 0, \tag{3.16}$$

or, equivalently

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \Delta t \sum_{j \in N(i)} \frac{s_{ij}}{v_i} \{A^-(\mathbf{U}^n_{Roe,(i,j)}) + D(\mathbf{U}^n_{diff,(i,j)})\}(\mathbf{U}_j^n - \mathbf{U}_i^n).$$

Implicit scheme

$$\frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i^n}{\Delta t} + \sum_{j \in N(i)} \frac{s_{ij}}{v_i} \{A^-(\mathbf{U}^{n+1}_{Roe,(i,j)}) + D(\mathbf{U}^{n+1}_{diff,(i,j)})\}(\mathbf{U}_j^{n+1} - \mathbf{U}_i^{n+1}) = 0. \tag{3.17}$$

We use the Newton method (Sec.1.4.3) to solve the system in the case of the implicit scheme. Then, by denoting $\delta \mathbf{U}^{k+1} = \mathbf{U}^{k+1} - \mathbf{U}^k$, the variation of the $k-th$ iteration that approximates the solution of time step $n+1$, we have a linear system of unknown $\delta \mathbf{U}^{k+1}$ to solve :

$$\begin{aligned} &\frac{\delta \mathbf{U}_i^{k+1}}{\Delta t} + \sum_j \frac{s_{ij}}{v_i} \left[A^-(\mathbf{U}^k_{Roe,(i,j)}) + D(\mathbf{U}^k_{diff,(i,j)})\right] \left(\delta \mathbf{U}_j^{k+1} - \delta \mathcal{U}_i^{k+1}\right) \\ &\qquad = -\frac{\mathbf{U}_i^k - \mathbf{U}_i^n}{\Delta t} - \sum_j \frac{s_{ij}}{v_i} \left[A^-(\mathbf{U}^k_{Roe,(i,j)}) + D(\mathbf{U}^k_{diff,(i,j)})\right] (\mathbf{U}_j^k - \mathbf{U}_i^k). \end{aligned} \tag{3.18}$$

Defining the unknown vector $\mathcal{U} = (\mathbf{U}_1, \ldots, \mathbf{U}_N)^t$, each Newton iteration for the computation of $\mathcal{U}$ at time step $n+1$ requires the numerical solution of the following linear system :

$$\mathcal{A}(\mathcal{U}^k) \delta \mathcal{U}^{k+1} = b(\mathcal{U}^n, \mathcal{U}^k). \tag{3.19}$$

3.1.3 Scaling Strategy

The larger the time step, the worse the condition number of the matrix $\mathcal{A}$ in (3.19). As a consequence, it is important to apply a preconditioner before solving the linear system. The most popular choice is the Incomplete LU factorization (later named ILU, see Benzi (2002) for more details). The error made by the approximate factorization using an ILU preconditioner depends on the size of the off diagonal coefficients of the matrix. For a better performance of the preconditioner, it is desirable that off diagonal entries of the matrix have small magnitudes.

As we are interested in convection dominated flows, the main contributions to the matrix $\mathcal{A}$ comes from the convective part of the equations through the matrix A^- in the equation 3.18 (we neglect D in first approximation). Unfortunately, the coefficients of the Roe matrix have very different magnitudes for low Mach number flows. Consequently, A^- and hence $\mathcal{A}$ have coefficients with very different magnitudes.

We are now going to detail a procedure that scales the matrix coefficients so that they have the same magnitude. The matrix A^- can be expressed using a complete eigenstructure decomposition of the Roe matrix : $A = \sum_k \lambda_k L^k \otimes R^k$. The three eigenvalues of the Roe matrix are $v_n + c$, v_n (multiplicity d), and $v_n - c$. As we are interested in flows at low Mach number, we can assume $\vec{v} \approx 0$ and in that case the eigenvalues of A become $\lambda^- = -c, \lambda_v = 0$, and $\lambda^+ = +c$. The right and left eigenvectors $R^\pm$ and $L^\pm$ associated to the sound waves are :

$$R^\pm = (1, \pm c\mathbf{n}, \frac{c^2}{\gamma - 1})^t, \qquad L^\pm = \frac{1}{2}(0, \pm\frac{1}{c}\mathbf{n}, \frac{\gamma - 1}{c^2})^t. \tag{3.20}$$

We have :

$$\begin{aligned} A^- &= -cL^- \otimes R^- \qquad \text{for the upwind scheme,} \\ A^- &= \frac{1}{2}(cL^+ \otimes R^+ - cL^- \otimes R^-) \text{ for the centered scheme.} \end{aligned}$$

One sees from (3.20) that the disequilibrium in A^- coefficients comes from the difference in the magnitude of the components of the left and right eigenvectors of A. If we multiply the local matrix A^- to the left (respectively to the right) by a diagonal matrix with the coefficients

$$d_{sca} = \begin{pmatrix} 1 & \mathbf{0} & 0 \\ 0 & c\mathbf{n} & 0 \\ 0 & \mathbf{0} & \frac{c^2}{\gamma-1} \end{pmatrix},$$

$$d_{sca}^{-1} = \begin{pmatrix} 1 & \mathbf{0} & 0 \\ 0 & \frac{1}{c}\mathbf{n} & 0 \\ 0 & \mathbf{0} & \frac{\gamma-1}{c^2}) \end{pmatrix},$$

where $\mathbf{n}$ is the unit normal vector, we obtain vectors and matrices with better balanced coefficients :

$$d_{sca}^{-1} R^\pm = (1, \pm\mathbf{n}, 1)^t, \qquad d_{sca} L^\pm = (0, \pm\mathbf{n}, 1)^t,$$

$$L^\pm \otimes R^\pm = \frac{1}{2} \begin{pmatrix} 0 & \mathbf{0} & 0 \\ \pm\frac{1}{c}\mathbf{n} & \mathbf{n} \otimes \mathbf{n} & \pm\frac{c}{\gamma-1}\mathbf{n} \\ \frac{\gamma-1}{c^2} & \pm\frac{\gamma-1}{c}\mathbf{n}^t & 1 \end{pmatrix},$$

$$d_{sca}L^{\pm} \otimes R^{\pm}d_{sca}^{-1} = \frac{1}{2}\begin{pmatrix} 0 & \mathbf{0} & 0 \\ \pm\mathbf{n} & \mathbf{n}\otimes\mathbf{n} & \pm\mathbf{n} \\ 1 & \pm\mathbf{n}^t & 1 \end{pmatrix}.$$

Hence it is possible to balance the coefficients of A^- using a similarity transformation. Any mesh can be associated with two diagonal matrices D_{sca} and D_{sca}^{-1} having the size of the mesh and containing the successive coefficients of the local matrices d_{sca} and d_{sca}^{-1}. Instead of solving system (3.19), one can rather solve :

$$\tilde{\mathcal{A}}\mathcal{V} = \tilde{b}, \tag{3.21}$$

where $\tilde{\mathcal{A}} = D_{sca}\mathcal{A}D_{sca}^{-1}$, $\mathcal{V} = D_{sca}\mathcal{U}$ and $\tilde{b} = D_{sca}b$. System (3.21) can be resolved more easily using an ILU preconditioner. Once the solution $\mathcal{V}$ is obtained we compute $D_{sca}^{-1}\mathcal{V}$ to obtain the original unknown vector $\mathcal{U}$.

3.1.4 Numerical Results

We implemented the scheme (3.7) in the C++ code ParaFlow and validated it on two test cases : Lid driven cavity and Detonation problem.

Lid Driven Cavity

Problem Description This test consists of a steady-state single-phase laminar flow in an entrained cavity. The flow domain is a square with the upper lid moving at a uniform horizontal velocity. The no-slip condition $\mathbf{v} = (0,0)$ is applied on the other walls of the cavity (see Fig. 3.1).

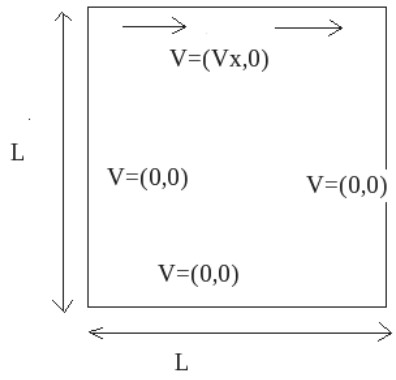

FIGURE 3.1 – *The computational domain*

The initial conditions are imposed as follows :
- The temperature $T = 35K$
- The dynamic viscosity $\nu = 0.025kg.m^{-1}.s^{-1}$
- The thermal conductivity $\lambda = 0.0027W.m^{-1}.K^{-1}$
- The pressure $P = 1$ bar.

The dymanic viscosity is chosen in order to have a Reynolds number equal to 400 with a horizontal velocity of the lid $v_x = 1m.s^{-1}$ (Re = $\rho v_x L/\nu$).

Explicit Scheme vs Implicit Scheme The respective merits of explicit and implicit schemes have been extensively discussed in the literature

(Pulliam (2011)). Here, we compare the two schemes in the solutions of the lid driven cavity test case. We first use an upwind explicit scheme to obtain the stationary state of the problem at time $t = t_{stat}$. We then use the upwind implicit scheme with different CFL numbers to obtain the solution at $t = t_{stat}$. We compare the solutions and computational times obtained with the two methods.

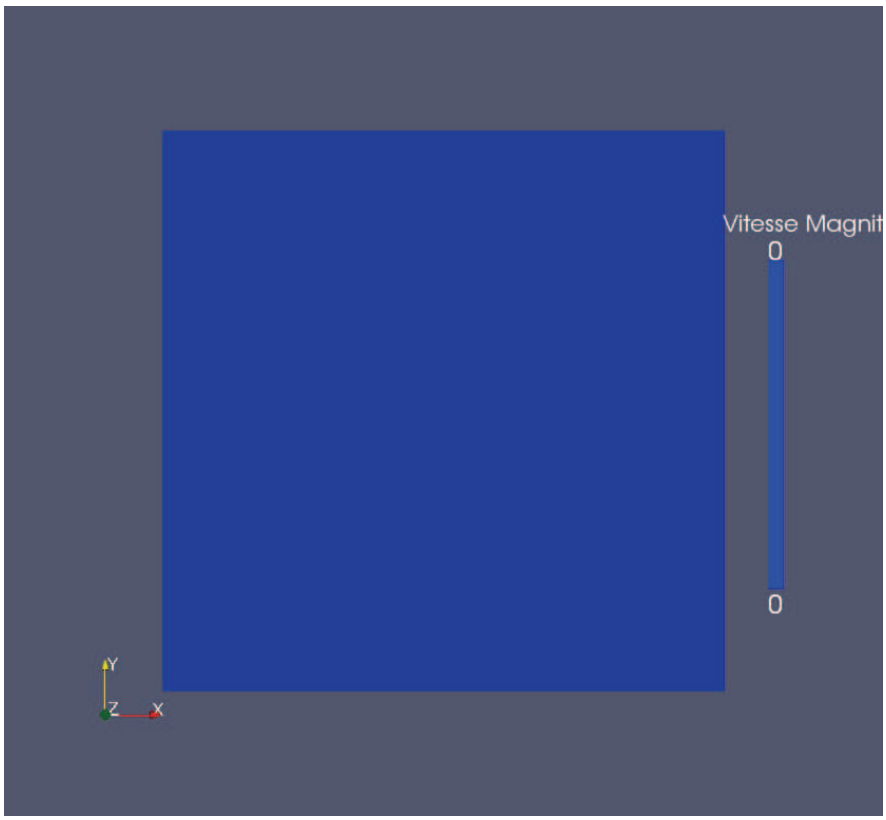

FIGURE 3.2 – *Initial state*

Fig. 3.3, 3.4, 3.5, 3.6, and 3.7 present the streamlines of the steady state result obtained using respectively the upwind explicit scheme and the upwind implicit scheme with CFL 100, 400, 800, and 1600. It can be seen that the solution of the implicit scheme with small CFL (100 , 400) is similar with that of explicit scheme. The solution of the implicit scheme with larger CFL (800 , 1600) is more diffusive than that of the explicit scheme. However, when we increase the number of time steps (4 instead of 2 for $CFL = 800$, 3 instead of 1 for $CFL = 1600$) of the implicit scheme, we obtain a solution similar to that of the explicit scheme.
Even though the implicit scheme is more diffusive than the explicit one, we see on practical cases that the implicit method has a much more reasonable computational time. Here, the explicit scheme needs 156 seconds to compute the steady state, while using the implicit with $CFL = 1600$ one needs less than 0.4 seconds. If one increases the number of cells for a better precision, the difference between the two families of schemes is more impressive.

Upwind Scheme vs Centered Scheme Figs. 3.8 and 3.9 present the streamlines of the steady state results obtained using either the upwind or

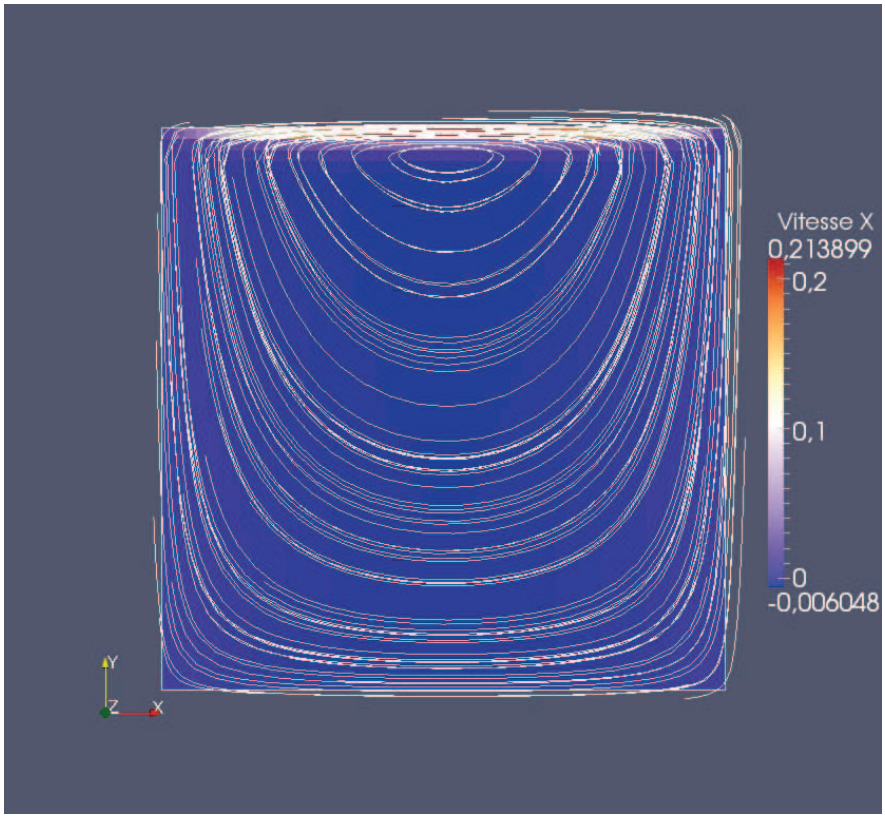

FIGURE 3.3 – *Steady State, Explicit scheme*

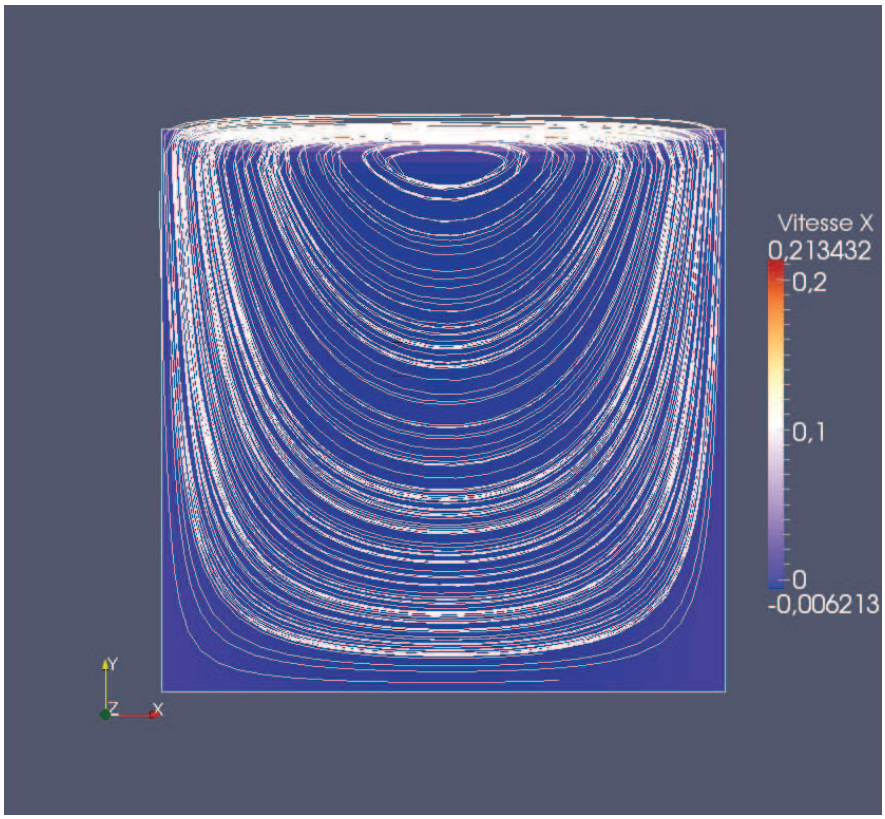

FIGURE 3.4 – *CFL* = 100, 16 *time steps*

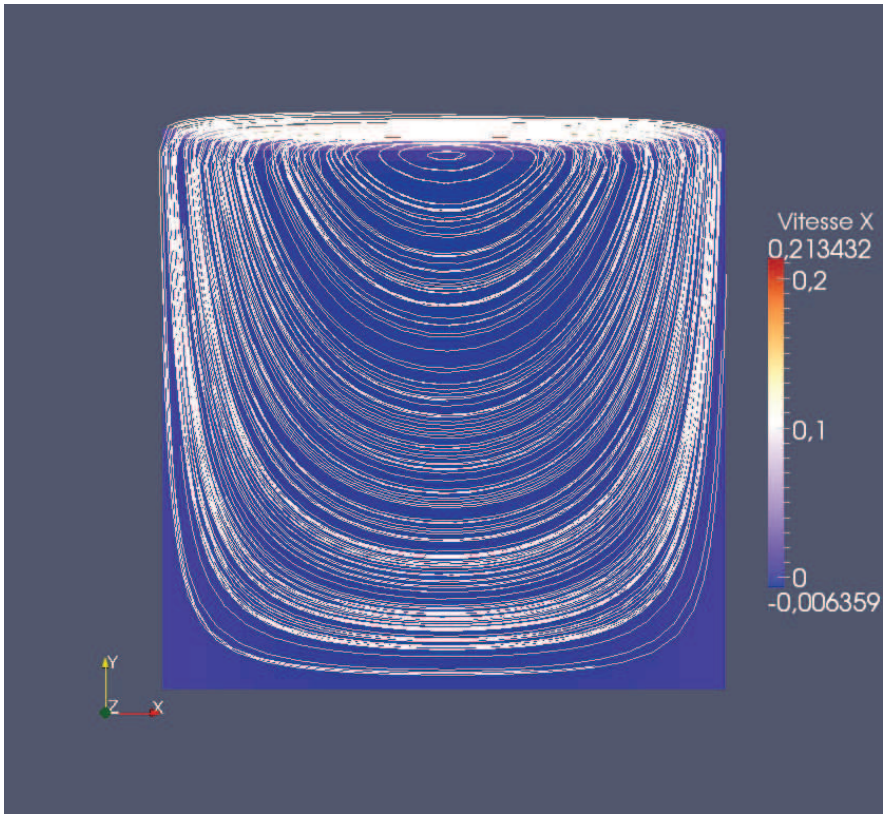

FIGURE 3.5 – *CFL* = 400, 4 *time steps*

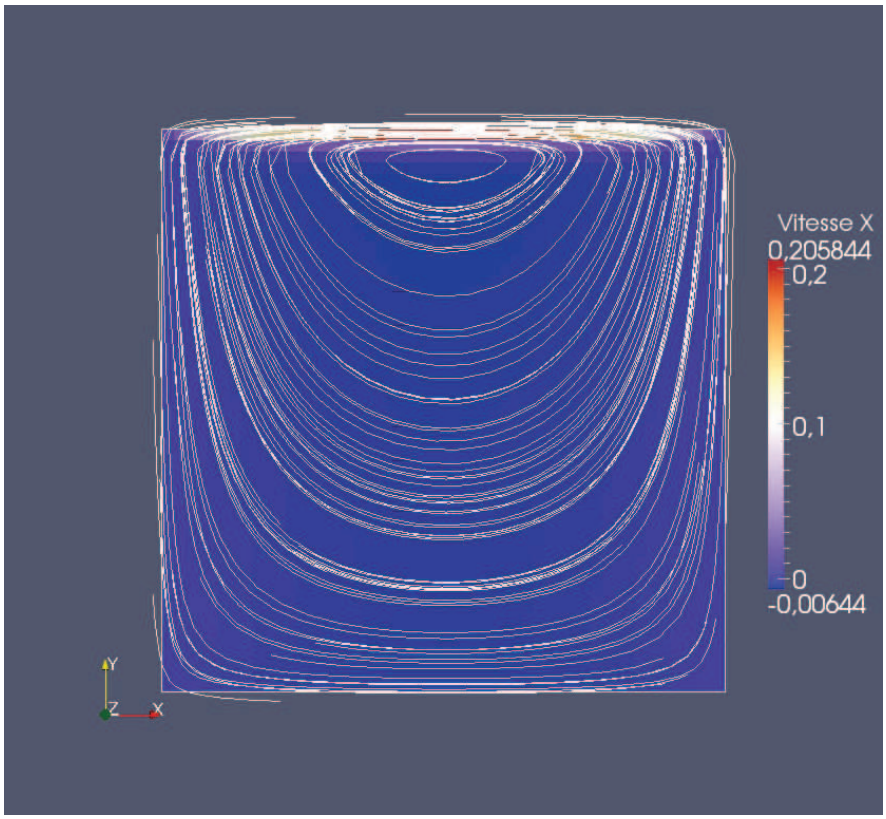

FIGURE 3.6 – *CFL* = 800, 2 *time steps*

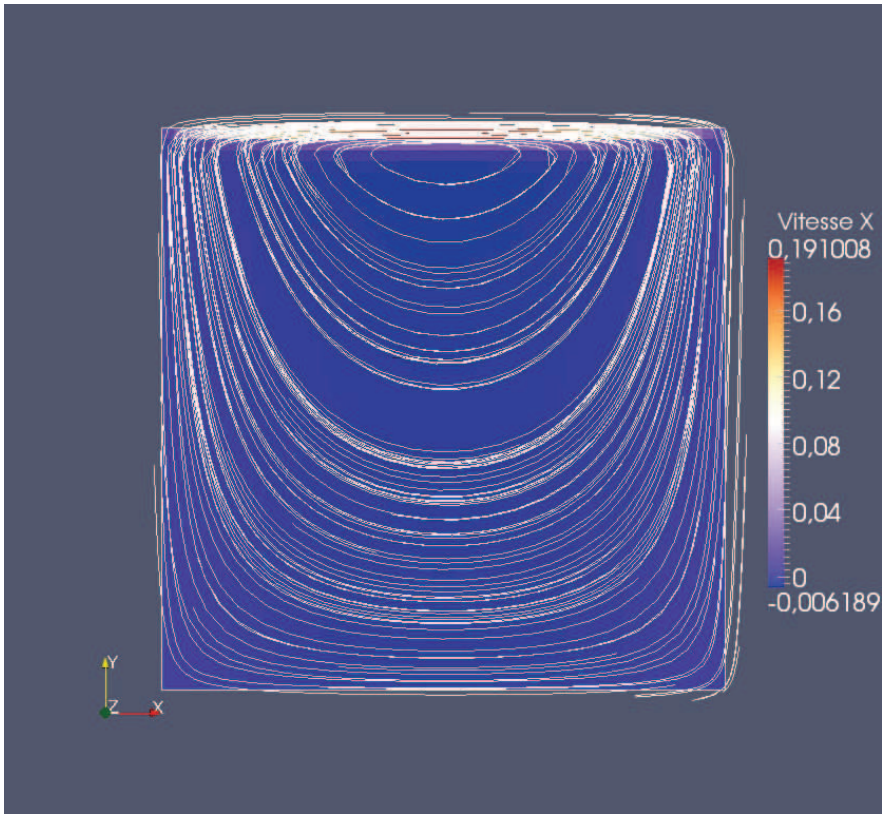

FIGURE 3.7 – *CFL* = 1600, 1 *time step*

CFL	0.5	100	400	800	1600
Number of time steps	3152	16	4	2	1
Time of computation	155.987	3.55	1.079	0.634	0.34

TABLE 3.1 – *Comparison of computational time, implicit vs explicit*

the centered schemes to discretize the convective part of the Navier–Stokes equations. Our test case is the same lid driven cavity test case presented in the previous section, solved on a cartesian 50×50 cell mesh. This case is a classical benchmark in the literature (see Ghia et al. (1982)) to validate numerical methods for the Navier-Stokes equations. The lid speed is $1\,m/s$, the maximum Mach number of the flow is 0.008. The Roe approximate Riemann solver (Roe (1981)) employed for the convection fluxes is known to have problem solving such low Mach number flows when the scheme is explicit, especially on multidimensional cartesian meshes (see Dellacherie (2010)). It can be seen on Fig. 3.8 that the upwind scheme does not capture the correct streamlines. However, on Fig. 3.9, it can be seen that the implicit centered scheme is much less diffusive and captures the correct solution with its expected three vortices.

Assessment of the Scaling strategy We now study the performance of our numerical methods on the same lid driven cavity test case presented in the previous section. In this section, we vary the time step (consequently the CFL number) and the mesh size. We also compare the direct solver with the iterative one and the effect of different preconditioners on the resolution of the linear systems.

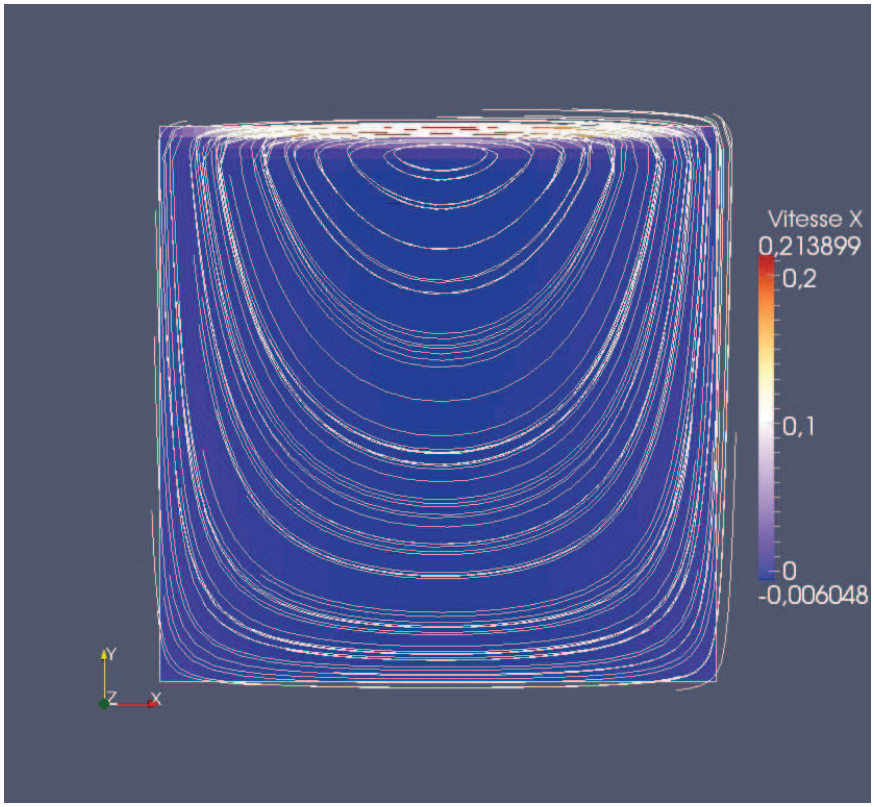

FIGURE 3.8 – *Steady state, upwind scheme*

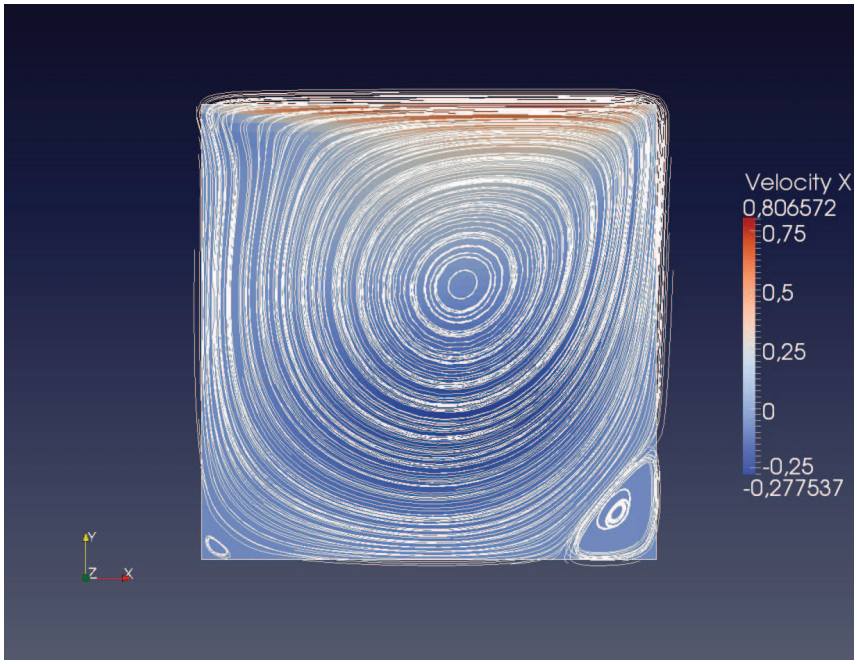

FIGURE 3.9 – *Steady state, centered scheme*

Considering first the upwind scheme, we remark that the ILU preconditioner with no level of fill-in performs well. Figs. 3.10 and 3.11 show the average number of GMRES iterations at each Newton iteration. We observe that the use of our Scaling strategy presented in Sect. 3.1.3 reduces more than twice the iteration numbers.

When we use the centered scheme, the system matrix has a poor diago-

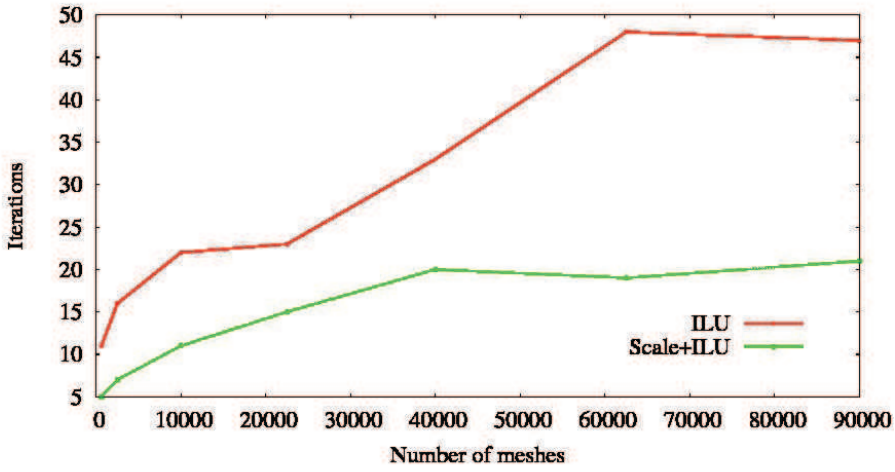

FIGURE 3.10 – *Number of GMRES iterations for the upwind scheme, CFL* 1000

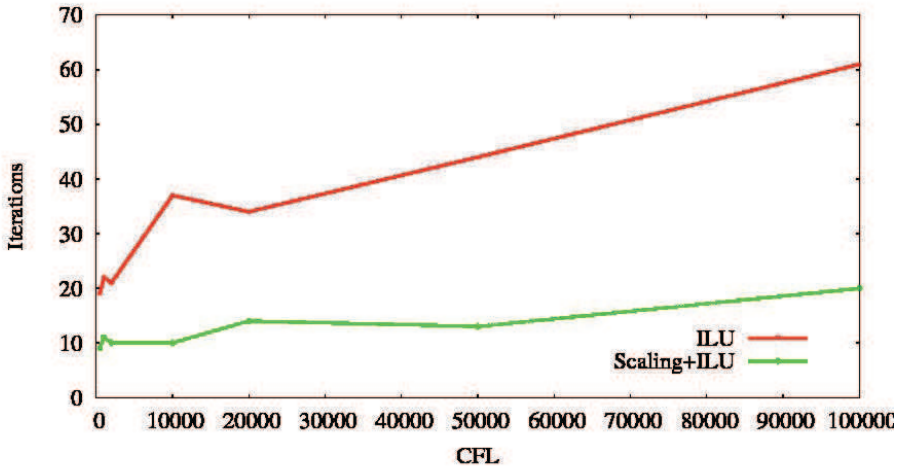

FIGURE 3.11 – *Number of GMRES iterations for the upwind scheme, mesh* 100×100

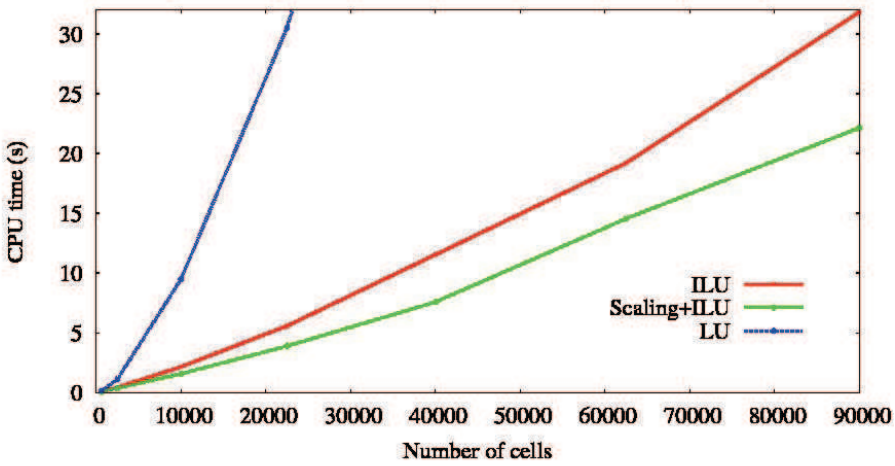

FIGURE 3.12 – *Computational time for the upwind scheme, CFL* 1000

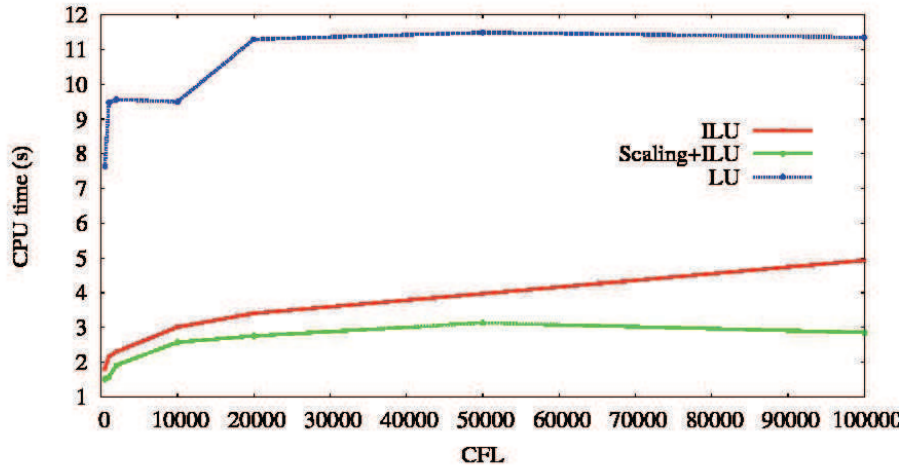

FIGURE 3.13 – *Computational time for the upwind scheme, mesh* 100×100

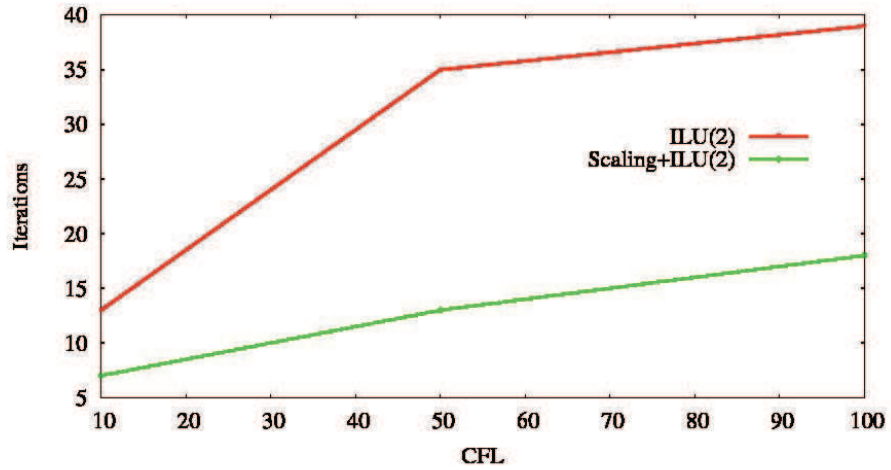

FIGURE 3.14 – *Number of GMRES iterations for the centered scheme, mesh* 50×50

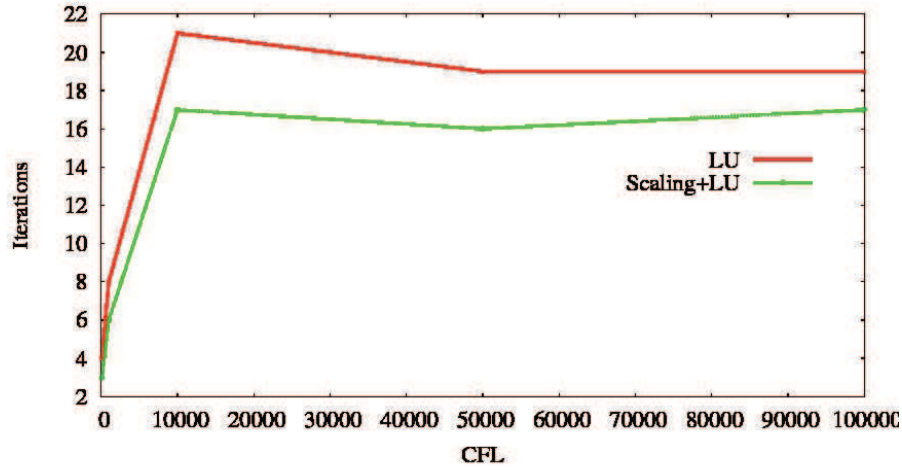

FIGURE 3.15 – *Number of Newton iterations for the centered scheme, mesh* 50×50

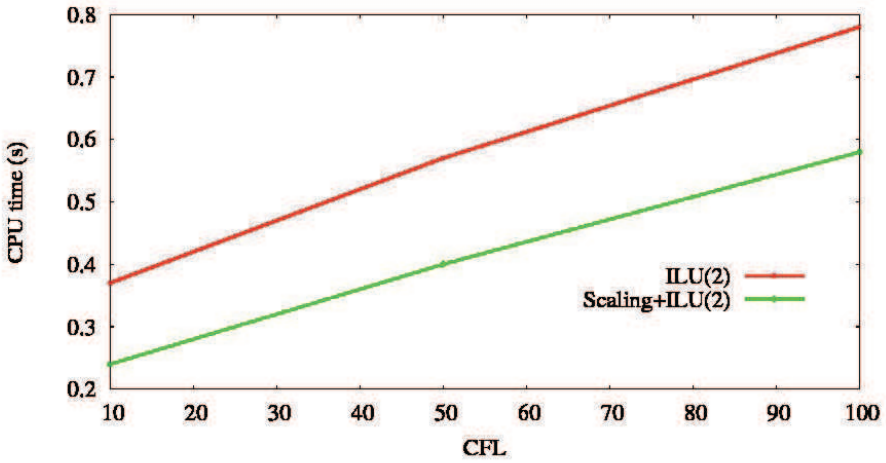

FIGURE 3.16 – *Computational time for the centered scheme, mesh* 50×50

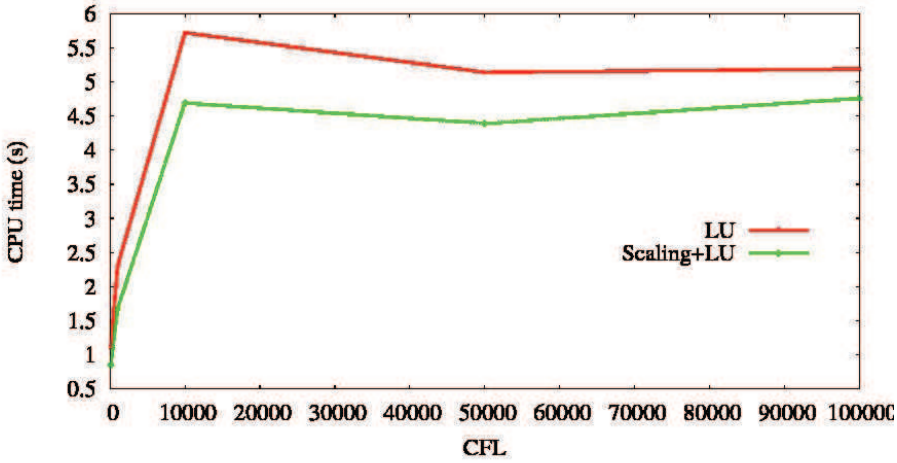

FIGURE 3.17 – *Computational time for the centered scheme, mesh* 50×50

nal, and ILU preconditioner with no fill-in is not efficient in preconditioning the linear system. One needs to use an incomplete factorisation with two levels of fill-in to solve the linear system up to the CFL 100, and the Scaling strategy enables to save a considerable number of iterations (Fig. 3.14). Beyond that value, only a direct solver is able to solve the system. However, one can remark that the Scaling strategy enables a reduction of the number of Newton iterations using a direct solver (Fig. 3.15). We also stress that the steady state solution obtained with very large CFL numbers is still accurate and displays the expected vortices.

Detonation Problem

Problem Description The problem consists of a pressurized ball of constant size (independent of the mesh) in a closed box at the initial state as presented in Fig.3.18. The pressure at the center of the box is imposed at 1.1 bar and the pressure outside the center is set to 1 bar. We then expect to see waves spread and reflect in the box. We neglect viscous and diffu-

sive terms in the model. We use this test case to check the conservative property of our scheme.

Numerical Solutions Figs. 3.19 and 3.20 show the profile of the pressure obtained using the upwind scheme to discretize the Euler equations (Eq. (3.1) with no viscosity and heat conductivity terms) on a cartesian 200x200 cell mesh. We use the fully implicit method with CFL = 10.

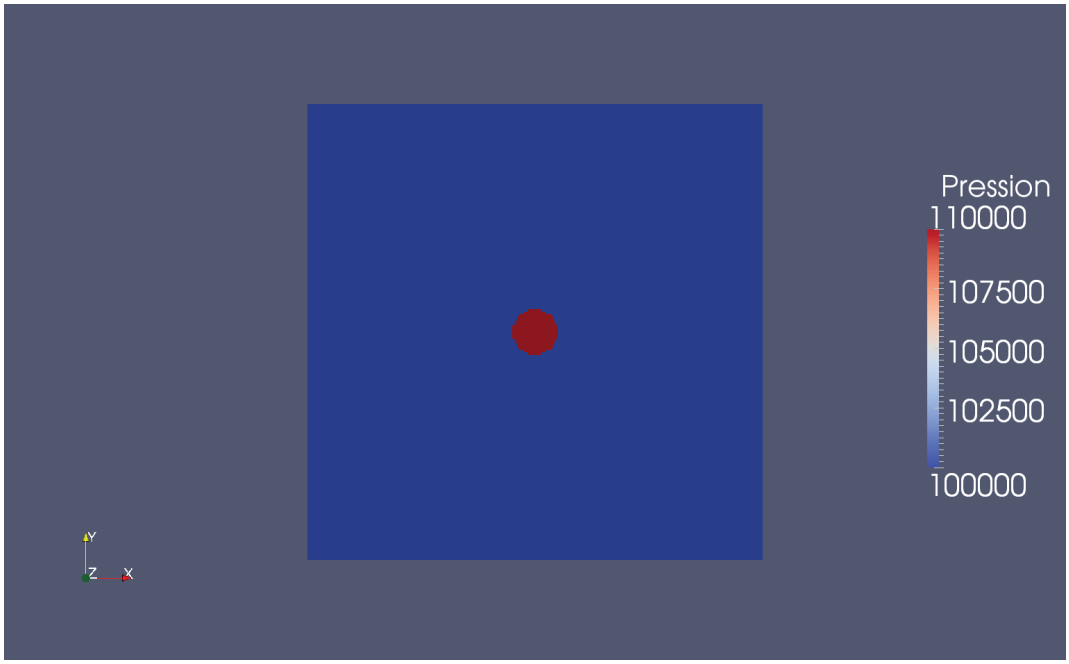

FIGURE 3.18 – *Initial state*

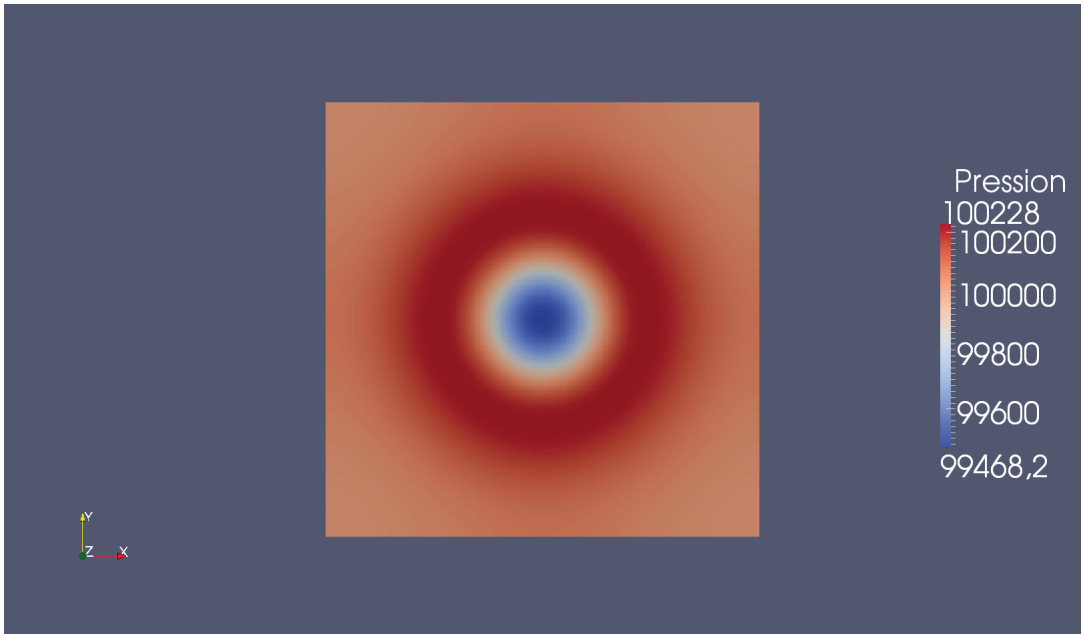

FIGURE 3.19 – *Profile of the pressure after* 5 *time steps*

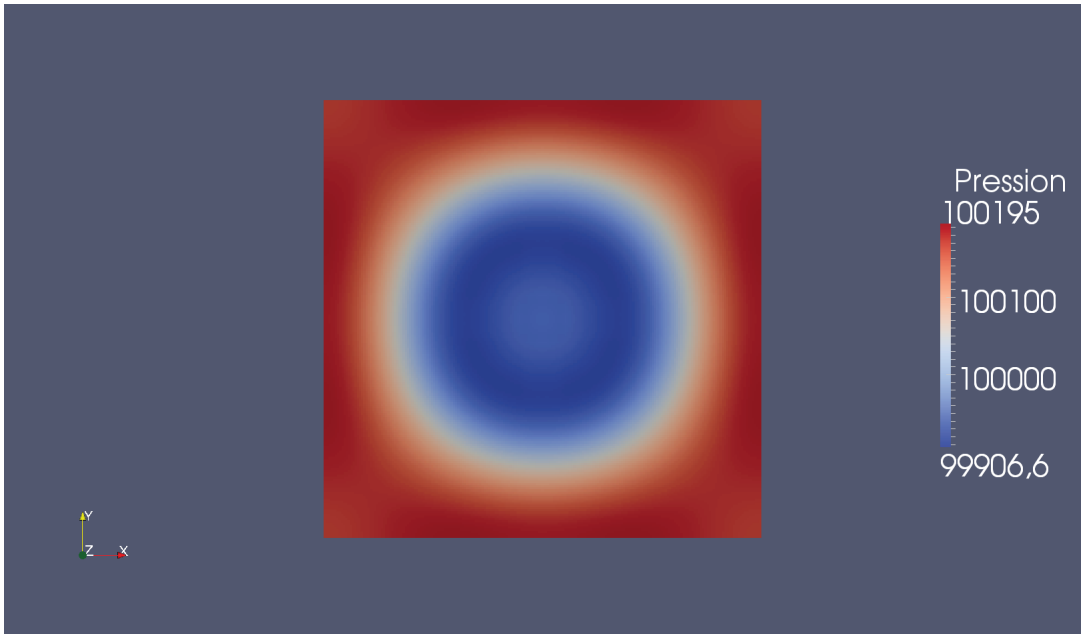

FIGURE 3.20 – *Profile of the pressure after* 10 *time steps*

Backward-facing Step

This test case is computed with FLICA-OVAP (Fillion et al. (2009)).

Problem Description This test consists of a steady-state single-phase laminar flow over a 2D backward-facing step. This case is also a well-known problem in the literature, with experiment in Armaly et al. (1983) and numerical results in Wan et al. (2002) and Zhu (1995). The initial conditions are set as follows :

- uniform velocity : $\mathbf{v} = (1.0, 0.0)m/s$,
- uniform pressure : $10^5 Pa$
- uniform enthapy : $4.10^5 J/kg$
- Reynolds number : 25

A uniform profile of velocity like in the initial conditions is imposed at the inlet, and a uniform profile of pressure like in the initial conditions is imposed at the outlet.

Upwind vs Centered Figs. 3.21 and 3.22 present the streamlines of the steady state result obtained using either the upwind or the centered schemes to discretize the convective part of the Navier–Stokes equations. We solved the problem on a cartesian 220×10 cell mesh. It can be seen on Fig. 3.21 that the upwind scheme does not capture the correct streamlines. However, on Fig. 3.22, it can be seen that the implicit centered scheme is much less diffusive and captures the correct solution (see Wan et al. (2002)) with its expected vortex.

Study of the Spectrum of the Matrices

It is well-known that the number of steps required for the convergence of a linear system $Ax = b$ to a satisfactory precision typically depends on

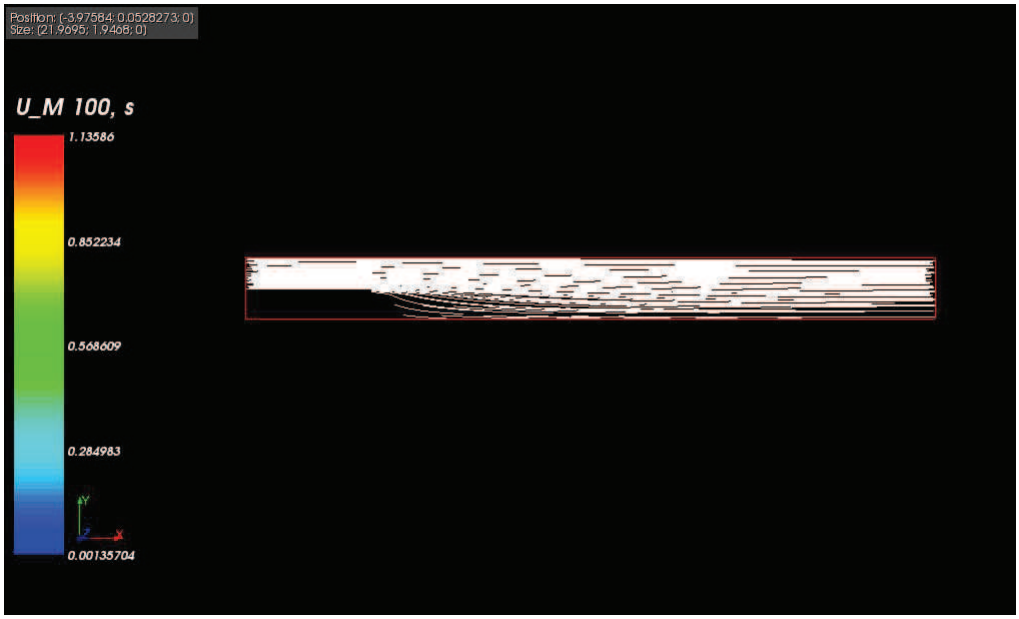

FIGURE 3.21 – *Explicit upwind scheme*

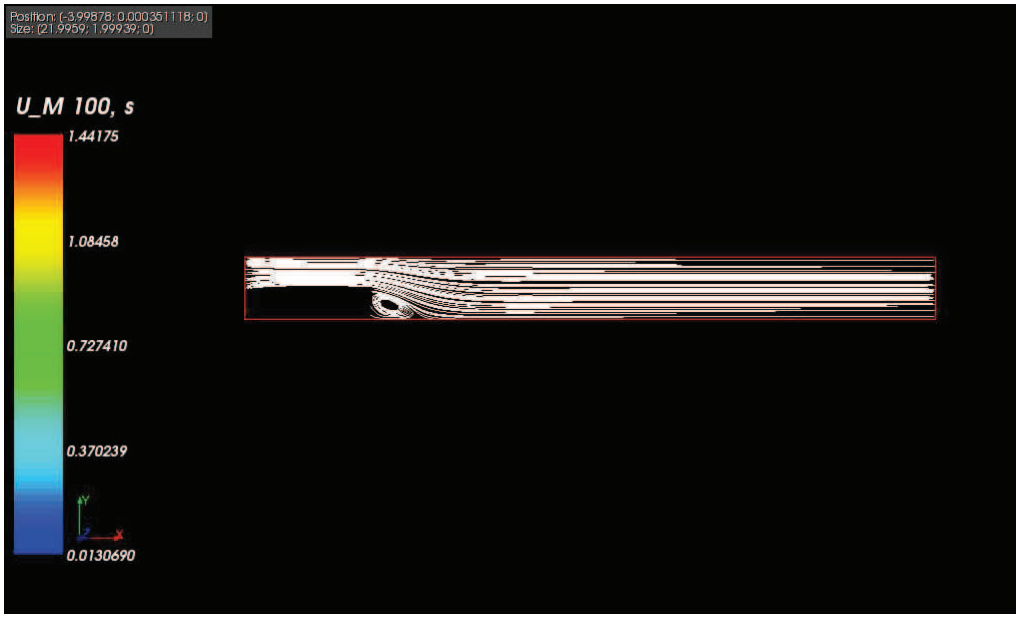

FIGURE 3.22 – *Implicit centered scheme*

the spectral properties of the matrix A. For example, the conjugate gradient iteration is guaranteed to solve a hermitian positive definite system quickly if the eigenvelues of A are clustered well away from the origin. In fact, for the conjugate gradient method, the appropriate approximation problem involves the A-norm and from Trefethen and D. Bau (1997), we have :

Theorem 3.1 *If the conjugate gradient iteration has not already converged before step m then we have*

$$\frac{\|e_m\|_A}{\|e_0\|_A} = \inf_{p \in P_m} \|p(A)e_0\|_A \leq \inf_{p \in P_m} \max_{\lambda \in \sigma(A)} |p(\lambda)|, \tag{3.22}$$

where e_m is the error at step m, P_m denotes the set of polynomials of degree at most m, and $\sigma(A)$ is the set of eigenvalues of A.

In the general case, where A is not positive definite and for the GMRES method, from Trefethen and D. Bau (1997), we have

$$\frac{\|r_m\|}{\|b\|} \leq \inf_{p \in P_m} \|p_m(A)\| \leq \kappa(V) \inf_{p \in P_m} \max_{\lambda \in \sigma(A)} |p(\lambda)|, \tag{3.23}$$

where P_m denotes the set of polynomials of degree at most m with $p(0) = 1$, V is a nonsingular matrix of eigenvectors of A (assuming A is diagonalizable), and $\sigma(A)$ is the set of all eigenvalues of A. Roughly speaking, this theorem says that fast convergence occurs when the eigenvalues of A are clustered away from the origin and A is not too far from normality.

We then study the spectrum of our matrices. As we mentioned in the previous sections, the larger the time step, the worse the condition number of the matrix A.

In Figs 3.23 , 3.24 and 3.25, we present the spectrum and the condition number of the matrix A using the upwind scheme on a cartesian 10×10 cell mesh respectively with CFL equal 1, 10 and 50. We can see that the conditon number increases with the CFL and the eigenvalues of A are clustered more closely to the origin. Fortunately, the classical preconditioner ILU works well for the upwind scheme matrix.

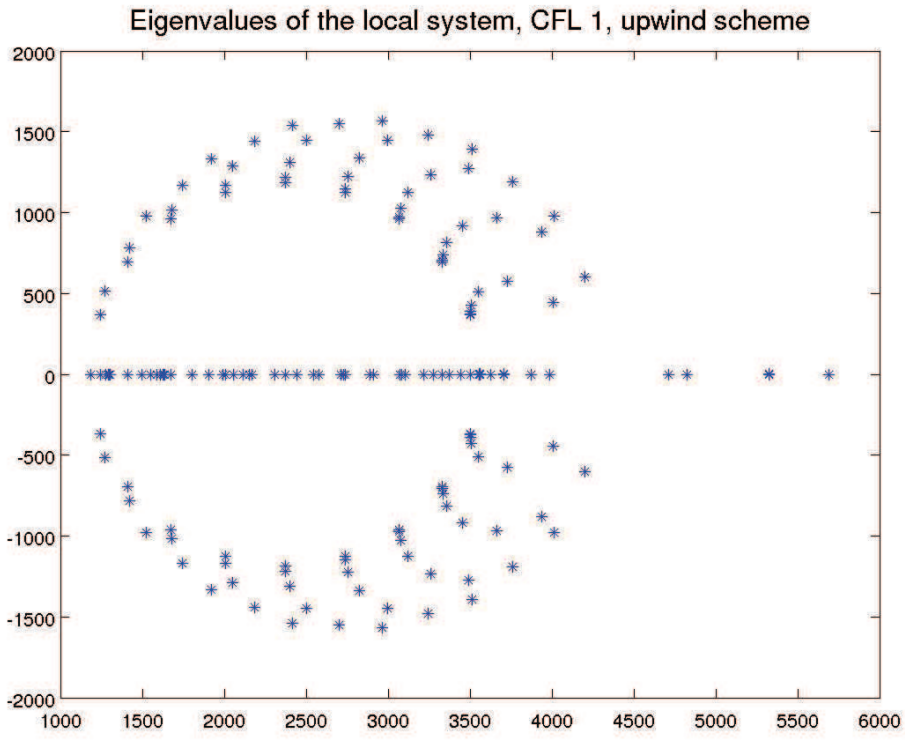

FIGURE 3.23 – *CFL* 1, 10×10 *cells, condition number* = $4.4e^4$

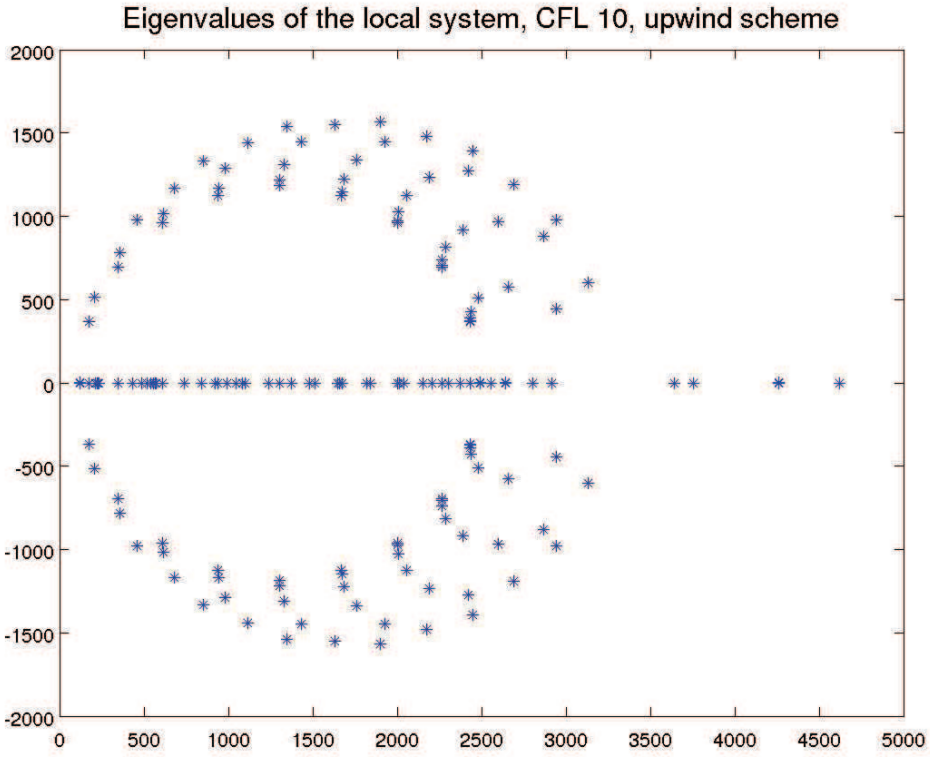

FIGURE 3.24 – *CFL* 10, 10×10 *cells, condition number* = $3.3e^5$

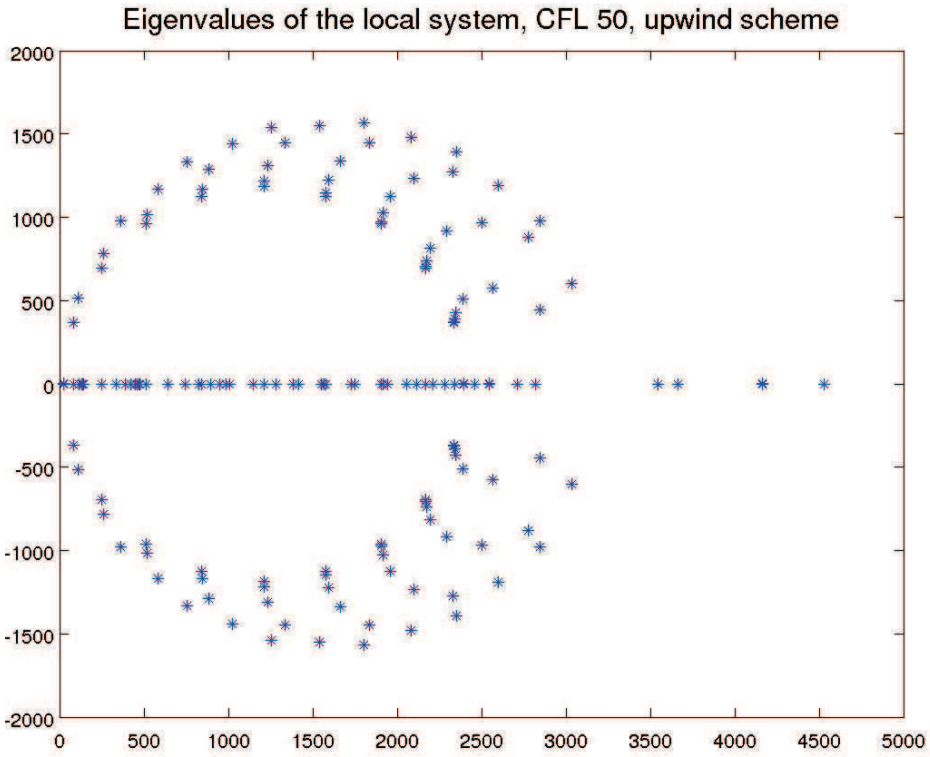

FIGURE 3.25 – *CFL* 50, 10×10 *cells, condition number* = $3.8e^5$

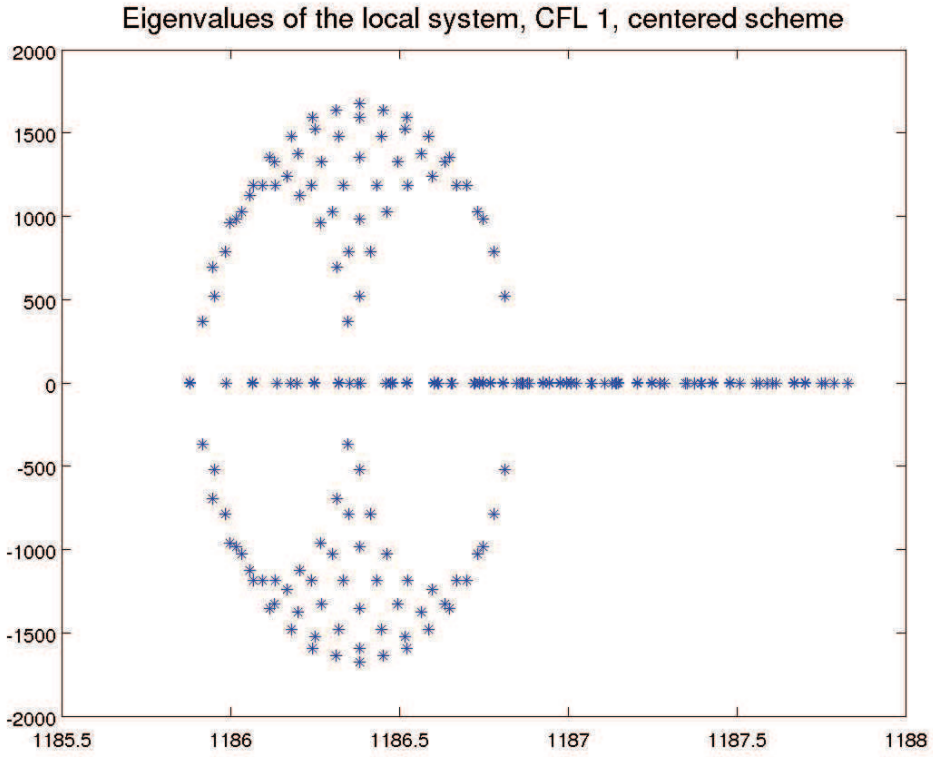

FIGURE 3.26 – *CFL* 1, 10×10 *cells, condition number* = $6.2e^4$

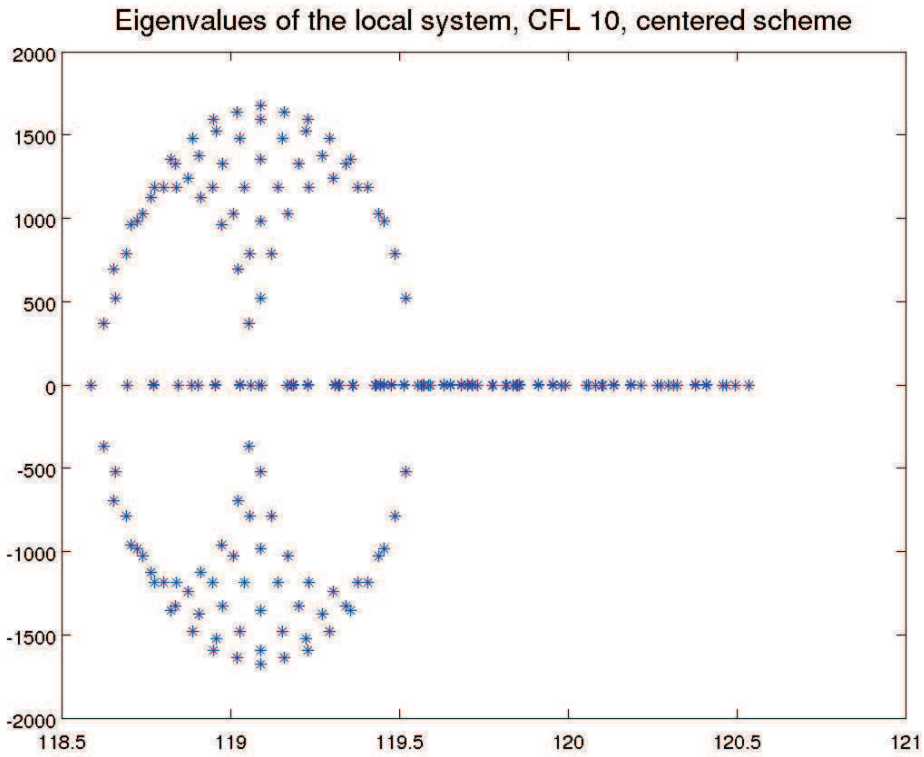

FIGURE 3.27 – *CFL* 10, 10×10 *cells, condition number* = $3.6e^5$

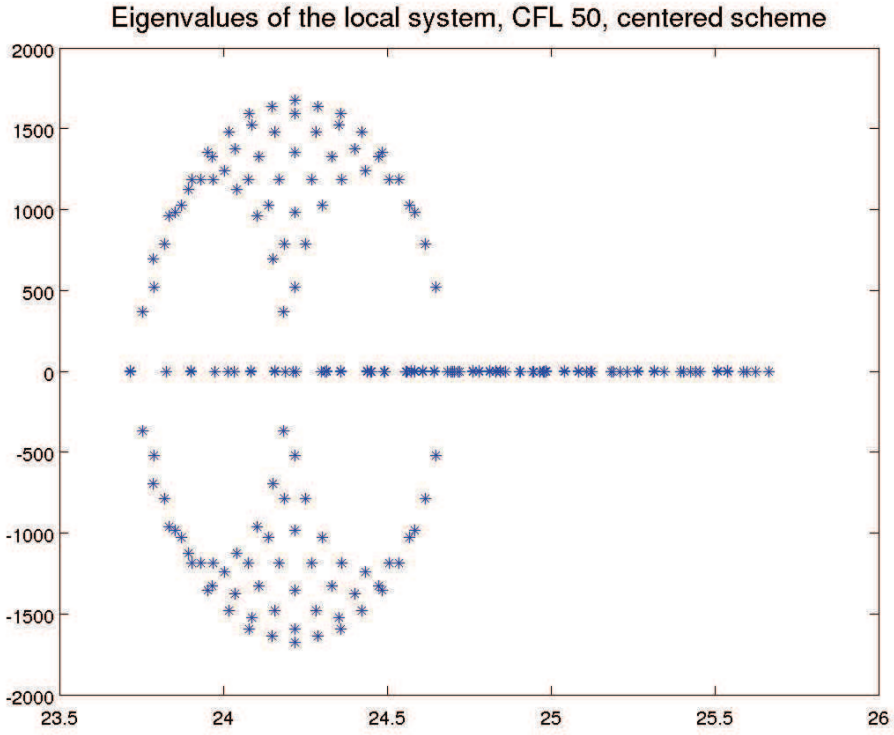

FIGURE 3.28 – *CFL* 50, 10×10 *cells, condition number* = $4e^5$

In Figs 3.26 , 3.27 and 3.28, we present the spectrum and the condition number of the matrix A using the centered scheme on a cartesian 10×10 cell mesh respectively with CFL equal 1, 10 and 50. We can see that the condition number increases with the CFL and the eigenvalues of A are clustered differently from those of the upwind scheme. This may be the reason why the classical preconditioner ILU does not performs well in the case of the centered scheme although the condition number of the two matrices is not very diffirent.

3.2 Two-phase Flows

3.2.1 Compressible Two-phase Flows

In fluid mechanics, two-phase flows occur in a system containing gas and liquid with a meniscus separating the two phases. The most distinguishing feature of a two-phase flow is the presence of interfaces separating the phases, and across which some of the fluid properties are discontinuous. The geometry of the interfaces has lead to a classification of the flow regimes : bubble, plug, slug, annular, etc. The description of the interfaces is a very difficult computational task because of the complex geometries and regime transitions. Depending on the problem at hand, the desired level of detail, and the computational resources available, there exists a range of techniques for the numerical simulation of two-phase flows. These techniques may be divided into three following categories :

- **Interface-tracking methods** The interfaces between two phases are fully resolved.
- **Particle-tracking methods** The bubbles or droplets are treated as point particles, and their indivivual trajectories are calculated.
- **Two-fluid methods** Macroscopic models have been developed that focus on the evolution of averaged quantities. No explicit information of the interface is retained.

The interface-tracking methods are generally the most expensive in computational time for a given physical domain. The two-fluid methods are less so, whereas the particle-tracking methods reside somewhere in between.

In this thesis, we study the two-fluid method for the simulation of two-phase flows, typically gas-liquid flows.

The two-fluid model equations for two-phase flows are obtained by averaging the balance equations for each separated phase, using space, time or ensemble averaged quantities (see Ishii (1975) and Drew and Passman (1999)). The unknown physical quantities are the volume fraction $\alpha_k \in [0,1]$, the density $\rho_k \geq 0$, and the velocity $\mathbf{u}_k$ of each phase. The subscript k stands for l if it is the liquid phase and g for the gas phase. The common averaged pressure of the two phases is denoted by p. In our model, pressure equilibrium between the two phases is postulated. For the sake of simplicity, we study the isentropic two-fluid model. This model can be written as follows :

$$\begin{cases} \frac{\partial(\alpha_g\rho_g)}{\partial t} & + & \nabla\cdot(\alpha_g\rho_g\mathbf{u}_g) & = 0 \\ \frac{\partial(\alpha_l\rho_l)}{\partial t} & + & \nabla\cdot(\alpha_l\rho_l\mathbf{u}_l) & = 0 \\ \frac{\partial(\alpha_g\rho_g\mathbf{u}_g)}{\partial t} & + & \nabla\cdot(\alpha_g\rho_g\mathbf{u}_g\otimes\mathbf{u}_g)+\alpha_g\nabla p & \\ & + & \Delta p\nabla\alpha_g-\nabla\cdot(\alpha_g\nu_g\nabla\mathbf{u}_g) & = \alpha_g\rho_g\mathbf{g} \\ \frac{\partial(\alpha_l\rho_l\mathbf{u}_l)}{\partial t} & + & \nabla\cdot(\alpha_l\rho_l\mathbf{u}_l\otimes\mathbf{u}_l)+\alpha_l\nabla p & \\ & + & \Delta p\nabla\alpha_l-\nabla\cdot(\alpha_l\nu_l\nabla\mathbf{u}_l) & = \alpha_l\rho_l\mathbf{g}, \end{cases} \tag{3.24}$$

with $\alpha_g+\alpha_l=1$, and the two equations of state(EOS) $\rho_g=\rho_g(p)$ and $\rho_l=\rho_l(p)$. In our problem, we use the stiffened equation of state. Here, $\mathbf{g}$ is the gravity, ν_k is the viscosity of phase k, and Δp denotes the pressure default $p-p_k$ between the bulk average pressure and the interfacial average pressure. In the following, we denote $\alpha := \alpha_g$ and so we have $\alpha_l = 1-\alpha$.

By denoting $m_k = \alpha_k \rho_k$, $\mathbf{q}_k = \alpha_k \rho_k \mathbf{u}_k$ the system (3.24) can be written as :

$$\begin{cases} \frac{\partial m_g}{\partial t} + \nabla \cdot \mathbf{q}_g = 0 \\ \frac{\partial (m_l)}{\partial t} + \nabla \cdot \mathbf{q}_l = 0 \\ \frac{\partial \mathbf{q}_g}{\partial t} + \nabla \cdot (\mathbf{q}_g \otimes \frac{\mathbf{q}_g}{m_g}) + \alpha_g \nabla p \\ \qquad + \Delta p \nabla \alpha_g - \nabla \cdot (\alpha_g \nu_g \nabla \frac{\mathbf{q}_g}{m_g}) = m_g \mathbf{g} \\ \frac{\partial \mathbf{q}_l}{\partial t} + \nabla \cdot (\mathbf{q}_l \otimes \frac{\mathbf{q}_l}{m_l}) + \alpha_l \nabla p \\ \qquad + \Delta p \nabla \alpha_l - \nabla \cdot (\alpha_l \nu_l \nabla \frac{\mathbf{q}_l}{m_l}) = m_l \mathbf{g}. \end{cases} \tag{3.25}$$

We will deal with this system from now on. Due to the term $\alpha_k \nabla p$, the system cannot be written in conservation form in terms of the variables m_k and $\mathbf{q}_k$. Therefore, special care is needed for the spatial discretization of the system.
In this system, m_k and $\mathbf{q}_k$ are the unknown functions to determine. The volume fraction α_k and the bulk average pressure p can be determined by the following function :

$$\begin{pmatrix} \alpha_g \\ p \end{pmatrix} = f^{-1} \begin{pmatrix} \alpha \rho_g \\ (1-\alpha)\rho_l \end{pmatrix},$$

where the function f is defined as below :

$$f \begin{pmatrix} \alpha \\ p \end{pmatrix} = \begin{pmatrix} \alpha \rho_g(p) \\ (1-\alpha)\rho_l(p) \end{pmatrix}. \tag{3.26}$$

It should be noted here that α and p are defined such that the two phases have the same pressure which means :

$$\rho_g^{-1}(\frac{m_g}{\alpha_g}) = \rho_l^{-1}(\frac{m_l}{\alpha_l}). \tag{3.27}$$

We also define the sound speed of each phase as

$$c_g = \left(\frac{\partial \rho_g}{\partial p}\right)^{-\frac{1}{2}}, \quad c_l = \left(\frac{\partial \rho_l}{\partial p}\right)^{-\frac{1}{2}}.$$

For the computation of the differential, we also need to introduce the parameter γ^2 which is defined as :

$$\gamma^2 = \frac{c_g^2 c_l^2}{\alpha_g \rho_l c_l^2 + \alpha_l \rho_g c_g^2}, \tag{3.28}$$

we obtain the differential form of the pressure and the volume fraction as follows :

$$\begin{aligned} \tfrac{1}{\gamma^2} \mathrm{d}p &= \rho_l \, \mathrm{d}m_g + \rho_g \, \mathrm{d}m_l, \\ \tfrac{1}{\gamma^2} \mathrm{d}\alpha_g &= \tfrac{\alpha_l}{c_l^2} \, \mathrm{d}m_g - \tfrac{\alpha_g}{c_g^2} \, \mathrm{d}m_l. \end{aligned} \tag{3.29}$$

In a similar way to the single phase flow, by denoting $\mathbf{U} = \begin{pmatrix} m_g \\ \mathbf{q}_g \\ m_l \\ \mathbf{q}_l \end{pmatrix}$ we can write the system (3.25) as follows :

$$\frac{\partial \mathbf{U}}{\partial t} + F^{conv}(\mathbf{U}) + F^{diff}(\mathbf{U}) = S(\mathbf{U}). \tag{3.30}$$

Here, $S(\mathbf{U})$ is a source term (in our case, we consider only the gravity), and $F^{conv}(\mathbf{U})$ is the inviscid term and $F^{diff}(\mathbf{U})$ is the viscous term of the system. One therefore has :

$$F^{conv}(\mathbf{U}) = \begin{pmatrix} \nabla \cdot \mathbf{q}_g \\ \nabla \cdot \mathbf{q}_l \\ \nabla \cdot (\mathbf{q}_g \otimes \frac{\mathbf{q}_g}{m_g}) + \alpha_g \nabla p + \Delta p \nabla \alpha_g \\ \nabla \cdot (\mathbf{q}_l \otimes \frac{\mathbf{q}_l}{m_l}) + \alpha_l \nabla p + \Delta p \nabla \alpha_l \end{pmatrix}, \tag{3.31}$$

$$F^{diff}(\mathbf{U}) = \begin{pmatrix} 0 \\ 0 \\ -\nabla \cdot (\alpha_g \nu_g \nabla \frac{\mathbf{q}_g}{m_g}) \\ -\nabla \cdot (\alpha_l \nu_l \nabla \frac{\mathbf{q}_l}{m_l}) \end{pmatrix}, \tag{3.32}$$

and the source term is given by :

$$S(\mathbf{U}) = \begin{pmatrix} 0 \\ 0 \\ m_g \mathbf{g} \\ m_l \mathbf{g} \end{pmatrix}. \tag{3.33}$$

Spectrum and Hyperbolicity of the Inviscid Part

For the sake of simplicity, we consider the one-dimensional model and exclude any source terms. The inviscid part of the two-phase flow can be written in the quasi-linear form :

$$\frac{\partial \mathbf{U}}{\partial t} + A(\mathbf{U}) \frac{\partial \mathbf{U}}{\partial x} = 0, \tag{3.34}$$

where the matrix A is given by

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \gamma^2 \left(\alpha_g \rho_l + \frac{\alpha_l}{c_l^2} \triangle p \right) - u_g^2 & 2u_g & \gamma^2 \left(\alpha_g \rho_g - \frac{\alpha_g}{c_g^2} \triangle p \right) & 0 \\ 0 & 0 & 0 & 1 \\ \gamma^2 \left(\alpha_l \rho_l - \frac{\alpha_l}{c_l^2} \triangle p \right) & 0 & \gamma^2 \left(\alpha_l \rho_g + \frac{\alpha_g}{c_g^2} \triangle p \right) - u_l^2 & 2u_l \end{pmatrix}.$$

We first want to study the hyperbolicity of the system 3.34. We look for

the roots of the characteristic polynomial P_A of A. A straightforward computation leads to the following polynomial

$$P_A(X) = (X-u_g)^2(X-u_l)^2 - \alpha_g\rho_l\gamma^2(X-u_l)^2 - \alpha_l\rho_g\gamma^2(X-u_g)^2 + \triangle p\gamma^2. \tag{3.35}$$

The detailed study of this polynomial (Ndjinga (2007b)) proves the importance of taking into account the interfacial pressure to obtain a well-posed problem.

To study the roots of P_A we use a perturbation method (Toumi et al. (1999)) by introducing the small ratio

$$\xi = \frac{u_r}{a_m}, \tag{3.36}$$

where $u_r = u_g - u_l$ is the relative velocity of the two fluids and a_m is the "characteristic" speed of sound of the mixture, given by

$$a_m = \left(\frac{\rho_m(\alpha_g\rho_l + \alpha_l\rho_g}{\rho_g\rho_l}\right), \rho_m = \alpha_g\rho_g + \alpha_l\rho_l. \tag{3.37}$$

The first order approximation of the two-fluid system eigenvalues gives

$$\begin{cases} \lambda_1 = \frac{\alpha_l\rho_g u_g + \alpha_g\rho_l u_l}{\alpha_g\rho_l + \alpha_l\rho_g} + \sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\triangle p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}u_r^2\right)} + \mathcal{O}(\xi^2) \\ \lambda_2 = \frac{\alpha_l\rho_g u_g + \alpha_g\rho_l u_l}{\alpha_g\rho_l + \alpha_l\rho_g} - \sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\triangle p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}u_r^2\right)} + \mathcal{O}(\xi^2) \\ \lambda_3 = \frac{\alpha_l\rho_g u_g + \alpha_g\rho_l u_l}{\alpha_g\rho_l + \alpha_l\rho_g} - c_m + \mathcal{O}(\xi^2) \\ \lambda_4 = \frac{\alpha_l\rho_g u_g + \alpha_g\rho_l u_l}{\alpha_g\rho_l + \alpha_l\rho_g} + c_m + \mathcal{O}(\xi^2). \end{cases} \tag{3.38}$$

We see from (3.38) that for small relative velocities, the two eigenvalues λ_3 and λ_4 are always real and have the order of magnitude of the "characteristic" speed of sound c_m. In contrast, if $\Delta p < \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}u_r^2$, the two other eigenvalues λ_1 and λ_2 are complex and the system (3.34) is not hyperbolic. Hence it is not a well-posed problem. In order to obtain real eigenvalues, it is neccessary to have :

$$\Delta p \geq \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l + \alpha_l\rho_g}u_r^2.$$

In the literature, several models for the interfacial pressure have been proposed. However, the physical content is often debatable.
Here, we use the following formula for the interfacial pressure :

$$\Delta p = 1.1\frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l + \alpha_l\rho_g}u_r^2.$$

Asymptotic Analysis of the Isentropic Two-phase Model We investigate the behavior of the isentropic two-phase model when the volume fraction of one of the phases vanishes. We consider, for instance, that the

vapor phase is disappearing which means $\alpha_g = 0$. In this limit case, we have :

$$\Delta p = 0, \quad \gamma^2 = \frac{c_l^2}{\rho_g}$$

and the matrix A becomes

$$A_{\alpha_g=0} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -u_g^2 & 2u_g & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{\rho_l c_l^2}{\rho_g} & 0 & c_l^2 - u_l^2 & 2u_l \end{pmatrix}$$

which has four real eigenvalues

$$u_g,\ u_g,\ u_l + c_l,\ u_l - c_l,$$

but the matrix is not diagonalizable because there are only three eigenvectors :

$$\begin{aligned} \vec{v}_{u_l \pm c_l} &= {}^t(0,\ 0,\ 1,\ u_l \pm c_l) \\ \vec{v}_{u_g} &= {}^t\left(\frac{\rho_g}{\rho_l}\left(\frac{(u_g - u_l)^2}{c_l^2} - 1\right),\ u_g \frac{\rho_g}{\rho_l}\left(\frac{(u_g - u_l)^2}{c_l^2} - 1\right),\ 1,\ u_g\right) \end{aligned}$$

In fact, the two equations of the vapor phase are idential to a pressureless gas dynamics system whose eigenvalues are double but whose jacobian matrix is not diagonalizable. We then lost the hyperbolicity when $\alpha_g = 0$.

3.2.2 Numerical Scheme

Most of the numerical methods used in two-phase flow computer codes are based upon semi-implicit finite difference schemes with staggered grids and donor-cell differencing. The main features of these schemes are their efficiency and their robustness. However, these methods have a large amount of numerical dissipation, giving poor accuracy in smooth regions of the flow. Moreover, discontinutities are heavily smeared on coarse grids and oscillations appear when the grid is refined (Toumi et al. (2000)).
Here, we propose to use an approximate Riemann solver to discretize and solve the system (3.25). Using the finite volume method described in Sec. 3.1.2, we obtain the discretized system of (3.25) as follows :

$$\int_{C_i} \frac{\partial \mathbf{U}}{\partial t}\, \mathrm{d}\mathbf{x} + \sum_{j \in N(i)} \Phi_{ij}^{conv} + \sum_{j \in N(i)} \Phi_{ij}^{diff} = \int_{C_i} S(\mathbf{U}) \mathrm{d}\mathbf{x}, \tag{3.39}$$

where Φ_{ij}^{conv}, Φ_{ij}^{diff} denote the numerical flux of convection and diffusion on the cell C_i in direction of the neighbor cell C_j.
The evaluation of the diffusive contributions requires a knowledge of the first derivatives of intensive variables as velocities. They are obtained by a centered difference scheme in which the derivatives are represented by piecewise constant functions over computational domain as we work with

Cartesian mesh. We use the fomula similar to the case of single phase flow to discretize the numerical flux of diffusion. We describe here the strategy to discretize the numerical flux of convection.
Due to the $\alpha_k \nabla p$ and $\Delta p \nabla \alpha_k$ terms, the inviscid part of the two-phase flow cannot be written in a conservative form. But this system (in the case of one-dimensional) can be written in the quasi-linear form :

$$\frac{\partial \mathbf{U}}{\partial t} + A(\mathbf{U}) \frac{\partial \mathbf{U}}{\partial x} = 0. \tag{3.40}$$

Under some assumptions, in Toumi and Kumbaro (1999) the authors were able to obtain a conservative form that allowed them to give a sense to discontinous solutions, and were able to develop an approximate Riemann solver of Roe-type for the system (3.40) providing a local linearization of the non-conservative term $\alpha_k \nabla p$. We can also construct other linearizations than that of Toumi and Kumbaro (1999). Here, we do not propose a specific linearization but a general method for the construction of the Roe matrix once we have chosen a linearization. We then define a local inviscid flux function $F^{loc}$and a local Roe matrix A_{Roe} for this linearization. The inviscid flux in the normal direction to the cell interface $\partial C_{i,j}$ is given by :

$$\begin{aligned} \Phi_{ij}^{conv} &= \frac{F^{loc}(\mathbf{U}_i) + F^{loc}(\mathbf{U}_j)}{2}.\mathbf{n}_{ij} + \mathcal{D}\frac{\mathbf{U}_i - \mathbf{U}_j}{2} \\ &= F^{loc}(\mathbf{U}_i).\mathbf{n}_{ij} + A^{-}(\mathbf{U}_j - \mathbf{U}_i), \end{aligned} \tag{3.41}$$

where $\mathcal{D}$ is an upwinding matrix, A_{Roe} the Roe matrix and $A^{\pm} = \frac{1}{2}(A_{Roe} \pm \mathcal{D})$. The choice $\mathcal{D} = 0$ gives the centered scheme, whereas $\mathcal{D} = |A_{Roe}|$ gives the upwind scheme.
In the following, we will describe a general method for the construction of the Roe matrice A_{Roe}.

3.2.3 Roe Matrix for Isentropic Two-fluid Model

In this section, we introduce the Roe matrix for our problem. As we mentioned before, the system is not a conservative one, there are only three conservation laws on the variables $(m_g, m_l, q_g + q_l)$. We have to restrict the distribution solution because neither α or p must be continuous. Same for Δp and α. It should be also noted that Δp depends on ρ_k, α_k, u_k. Due to non-conservative terms, we do not seek a weak solution but a solution almost everywhere satisfing three conservation laws.
Under some simplifying assumptions ($\delta p = 0$, $c_l = \infty$, $1D$ flow) Toumi and Kumbaro (1999) were able to obtain a conservative form that allowed them to give a sense to discontinous solutions. It was also under those assumptions that they have been able to develop an approximate Riemann solver of Roe-type for the system (3.34) providing a local linearization of the non-conservative term $\alpha \frac{\partial p}{\partial x}$. Those local linearizations are given as follows :

$$\frac{1}{\overline{\alpha}_l} = \frac{1}{2}\left[\frac{1}{\alpha_l^l} + \frac{1}{\alpha_l^r}\right], \tag{3.42}$$

$$\overline{p} = \frac{\alpha_l^l p^l + \alpha_l^r p^r}{\alpha_l^l + \alpha_l^r}, \tag{3.43}$$

where ^{l}, ^{r} stand for left and right.
We can also construct other linearizations than that of Toumi and Kumbaro (1999). In this section, we will not propose a specific linearization but a general method for constructing the matrix of Roe, once we have chosen a linearization. Our goal is to develop a finite volume scheme which is consistent with the system (3.34) and satisfies three conservation laws at the discrete level. Once we have known the expression of the Roe matrix, we can develop a linearization which satisfies some non trival properties such as the positivity of the variable α.
Consider an initial condition which consists of two constant states :

$$U^l = \begin{pmatrix} m_g^l \\ q_g^l \\ m_l^l \\ q_l^l \end{pmatrix}, \qquad U^r = \begin{pmatrix} m_g^r \\ q_g^r \\ m_l^r \\ q_l^r \end{pmatrix}. \tag{3.44}$$

We need to solve the Riemann problem associated with the system (3.34) and the initial condition (3.44). Due to non-conservative terms, we do not seek a weak solution but a solution almost everywhere which satisfies three conservation laws. There are then more than one solution (three jump relations for four variables). Therefore, the solution depends on the way of linearizing locally the non-conservative terms. We are interested in the linearizations which consist in solving the following system of conservation laws

$$\begin{aligned} &\frac{\partial m_g}{\partial t} + \frac{\partial q_g}{\partial x} = 0 \\ &\frac{\partial m_l}{\partial t} + \frac{\partial q_l}{\partial x} = 0 \\ &\frac{\partial q_g}{\partial t} + \frac{\partial}{\partial x}\frac{q_g^2}{m_g} + \overline{\alpha}_g \frac{\partial p}{\partial x} + \overline{\triangle} p \frac{\partial \alpha_g}{\partial x} = 0 \\ &\frac{\partial q_l}{\partial t} + \frac{\partial}{\partial x}\frac{q_l^2}{m_l} + \overline{\alpha}_l \frac{\partial p}{\partial x} + \overline{\triangle} p \frac{\partial \alpha_l}{\partial x} = 0, \end{aligned} \tag{3.45}$$

where $\overline{\alpha}_k$ and $\overline{\triangle} p$ are the functions of U^L, U^R and satisfy

$$\begin{cases} \overline{\alpha}_g + \overline{\alpha}_l = 1 \\ \overline{\alpha}_k(U,U) = \alpha_k(U) \\ \overline{\triangle} p(U,U) = \triangle p(U). \end{cases} \tag{3.46}$$

The system (3.45) is a system of conservation laws which has the flux

$$F^{loc} : \begin{pmatrix} m_g \\ q_g \\ m_l \\ q_l \end{pmatrix} \to \begin{pmatrix} q_g \\ \frac{q_g^2}{m_g} + \overline{\alpha}_g p + \overline{\triangle} p\, \alpha_g \\ q_l \\ \frac{q_l^2}{m_l} + \overline{\alpha}_l p + \overline{\triangle} p\, \alpha_l \end{pmatrix}.$$

So for each linearization of $\overline{\alpha}_k, \overline{\triangle} p$, we need to find a Roe matrix i.e. a diagonalizable matrix $A^{Roe}(U^l, U^r)$ which satisfies

$$F^{loc}(U^l) - F^{loc}(U^r) = A^{Roe}(U^l, U^r)(U^l - U^r), \tag{3.47}$$

$$A^{Roe}(U,U) = \nabla F^{loc}(U). \tag{3.48}$$

We want to write the derivative of the flux

$$\Delta_l^r F^{loc} = F^{loc}(U^l) - F^{loc}(U^r) = \begin{pmatrix} \Delta_l^r q_g \\ \Delta_l^r \frac{q_g^2}{m_g} + \overline{\alpha}_g \Delta_l^r p + \overline{\triangle} p \Delta_l^r \alpha_g \\ \Delta_l^r q_l \\ \Delta_l^r \frac{q_l^2}{m_l} + \overline{\alpha}_g \Delta_l^r p + \overline{\triangle} p \Delta_l^r \alpha_l \end{pmatrix},$$

as a function of $\Delta_l^r U = U^L - U^R$.
We can write $\Delta_l^r q_k$ and $\Delta_l^r \frac{q_k^2}{m_k}$ as a function of $\Delta_l^r m_k$ and $\Delta_l^r q_k$ using the Roe phasic velocities

$$\overline{u}_k = \frac{\sqrt{m_k^l} u_k^l + \sqrt{m_k^r} u_k^r}{\sqrt{m_k^l} + \sqrt{m_k^r}}. \tag{3.49}$$

This expression does not admit a singularity since we never have $m_g = m_l = 0$ at the same time. Unfortunately, it is more difficult to write the jump $\Delta_l^r p$ and $\Delta_l^r \alpha$ as a function of $\Delta_l^r U$ which must be consistent with the equation (3.48).

Computation of Roe matrix : Jump of Pressure To compute the jump of pressure, we first write $\Delta_l^r m_k$ as a function of $\Delta_l^r p$ and $\Delta_l^r \alpha_k$. This expression can be written as

$$\begin{aligned} \Delta_l^r m_k &= {\alpha_k}^l {\rho_k}^l - {\alpha_k}^r {\rho_k}^r \\ &= {\alpha_k}^l ({\rho_k}^l - {\rho_k}^r) + {\rho_k}^r ({\alpha_k}^l - {\alpha_k}^r) \\ &= {\alpha_k}^l \Delta_l^r \rho_k + {\rho_k}^r \Delta_l^r \alpha_k \\ &= \frac{{\alpha_k}^l}{\overline{c}_k^2} \Delta_l^r p + {\rho_k}^r \Delta_l^r \alpha_k, \end{aligned}$$

with $\overline{c}_k^2 = \frac{p^l - p^r}{\rho_k^l - \rho_k^r}$. Since the function $\rho \to p(\rho)$ is increasing, we have $\overline{c}_k^2 \geq 0$.
Then to obtain Δp and $\Delta \alpha$ we need to solve the following system :

$$\begin{aligned} \frac{{\alpha_g}^l}{\overline{c}_g^2} \Delta_l^r p + {\rho_g}^r \Delta_l^r \alpha_g &= \Delta_l^r m_g, \\ \frac{{\alpha_g}^l}{\overline{c}_l^2} \Delta_l^r p + {\rho_g}^r \Delta_l^r \alpha_l &= \Delta_l^r m_l. \end{aligned}$$

Like in the continuous case (3.29), this system has the following solution

$$\begin{cases} \frac{1}{\overline{\gamma}^2} \Delta_l^r p = {\rho_l}^r \Delta_l^r m_g + {\rho_g}^r \Delta_l^r m_l \\ \frac{1}{\overline{\gamma}^2} \Delta_l^r \alpha_g = \frac{{\alpha_l}^l}{c_l^2} \Delta_l^r m_g - \frac{{\alpha_g}^l}{c_g^2} \Delta_l^r m_l, \end{cases} \tag{3.50}$$

where $\overline{\gamma}$ is defined similarly as in (3.28) :

$$\overline{\gamma}^2 = \frac{\overline{c}_g^2 \overline{c}_l^2}{{\alpha_g}^l {\rho_l}^r \overline{c}_l^2 + {\alpha_l}^l {\rho_g}^r \overline{c}_g^2}. \tag{3.51}$$

Since $\overline{c}_k^2 \geq 0$, we can conclude that $\overline{\gamma} \geq 0$.
Then a matrix which satisfies the conditions (3.47-3.48) can be written as

follows

$$A^{Roe} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \overline{\alpha}_g {\rho_l}^r \overline{\gamma}^2 + \frac{\alpha_l^l}{\overline{c}_l^2}\overline{\gamma}^2 \overline{\triangle p} - \overline{u}_g^2 & 2\overline{u}_g & \overline{\alpha}_g {\rho_g}^r \overline{\gamma}^2 - \frac{\alpha_g^l}{\overline{c}_g^2}\overline{\gamma}^2 \triangle p & 0 \\ 0 & 0 & 0 & 1 \\ \overline{\alpha}_l {\rho_l}^r \overline{\gamma}^2 - \frac{\alpha_l^l}{\overline{c}_l^2}\overline{\gamma}^2 \overline{\triangle p} & 0 & \overline{\alpha}_l {\rho_g}^r \overline{\gamma}^2 + \frac{\alpha_g^l}{\overline{c}_g^2}\overline{\gamma}^2 \overline{\triangle p} - \overline{u}_l^2 & 2\overline{u}_l \end{pmatrix}.$$

The characteristic polynomial of that matrix has the form as in (3.35). So we can conclude that A^{Roe} is diagonalizable with real eigenvalues if $\overline{\Delta p}$ is large enough.
This method can be extended to non-isentropic multiphase flows. But here for simplicity, we restrict ourselves to the isentropic two-phase flow.

3.2.4 The Entropy Correction

In 1.4.5 we saw that the Roe solver is not entropic and that we may need an entropic correction. As in the case of cross flows, λ_1 and λ_2 in (3.38) may change sign and in this case the Roe scheme can capture non entropic weak solution. We therefore propose an entropy fix for the Roe scheme. Our entropic Roe scheme consists of an upwingding matrix $D_{entropic} \geq D_{upwind}$, with a slightly larger diffusion on the characteristics that changes sign, meant to solve the entropy violation problem. Consider the Riemann problem between two states U_L and U_R and a characteristic λ taking values λ_L for U_L and λ_R for U_R. The corresponding eigenvalue of the Roe matrix $A(U_L, U_R)$ is λ_{LR}, and for $D_{entropic}$ it is

$$\lambda_{LR}^{entropic} = |\lambda_{LR}| + \frac{|\lambda_L - \lambda_R|}{2}.$$

We analyse the void waves structure in the next section to obtain more information.

3.2.5 Incompressible Limit of Two-phase Flows

In this section we present a derivation of the incompressible model of Keyfitz et al. (2003). In that article, the authors studied the non hyperbolicity of the system, and proposed to solve Riemann problems for the nonlinear, nonhyperbolic systems. They introduced the notion of singular shocks and the Riemann solutions found using those shocks have a reasonable physical interpretation. In this section, we use another approach to study the incompressible model. In fact, we add the interface pressure (as in the compresible case) and investigate the Riemann problem of that model.

We consider the one dimensional isentropic two-fluid system

$$\begin{cases} \frac{\partial(\alpha_g\rho_g)}{\partial t} + \frac{\partial(\alpha_g\rho_g u_g)}{\partial x} = 0, \\ \frac{\partial(\alpha_l\rho_l)}{\partial t} + \frac{\partial(\alpha_l\rho_l u_l)}{\partial x} = 0, \\ \frac{\partial(\alpha_g\rho_g \mathbf{u}_g)}{\partial t} + \frac{\partial(\alpha_g\rho_g {u_g}^2)}{\partial x} + \alpha_g\partial p + \Delta p\frac{\partial\alpha_g}{\partial x} = 0, \\ \frac{\partial(\alpha_l\rho_l \mathbf{u}_l)}{\partial t} + \frac{\partial(\alpha_l\rho_l {u_l}^2)}{\partial x} + \alpha_l\partial p + \Delta p\frac{\partial\alpha_l}{\partial x} = 0. \end{cases} \tag{3.52}$$

Assume the two phases are incompressible, we obtain the following system :

$$\partial_t\alpha_k + \partial_x(\alpha_k u_k) = 0, \tag{3.53}$$

$$\partial_t(\alpha_k\rho_i u_k) + \partial_x(\alpha_k\rho_k u_k^2) + \alpha_k\partial_x p + \Delta p\partial_x\alpha_k = 0, \tag{3.54}$$

where

$$\alpha_g + \alpha_l = 1.$$

By assuming smooth solutions we can replace (3.54) by :

$$\alpha_k\rho_k(\partial_t u_k + u_k\partial_x u_k) + \alpha_k\partial_x P + \Delta p\partial_x\alpha_k = 0. \tag{3.55}$$

Thanks to (3.53), we can deduce that :

$$\partial_x(\alpha_g u_g + \alpha_l u_l) = 0.$$

So $\alpha_g u_g + \alpha_l u_l$ is time-dependent but space independent. Thus we define

$$\alpha_g u_g + \alpha_l u_l = K(t). \tag{3.56}$$

Then K depends on the boundary condition. If we assume that the flow conditions are constant at one point, an inflow boundary for example, then we can say that K is a constant.

From Keyfitz et al. (2003), we introduce new variables :

$$\beta = \alpha_g\rho_l + \alpha_l\rho_g, \tag{3.57}$$

$$\omega = \rho_g u_g - \rho_l u_l. \tag{3.58}$$

The system (3.53) and (3.55) can be written as

$$\partial_t(\alpha_g\rho_l + \alpha_l\rho_g) + \partial_x(\alpha_g\rho_l u_g + \alpha_l\rho_g u_l) = 0, \tag{3.59}$$

$$\alpha_g\alpha_l\{\partial_t(\rho_g u_g - \rho_l u_l) + \partial_x(\rho_g\frac{u_g^2}{2} - \rho_l\frac{u_l^2}{2}) + \Delta p\partial_x\alpha_g = 0. \tag{3.60}$$

We have

$$\alpha_g = -\frac{\beta - \rho_g}{\rho_g - \rho_l}, \quad \alpha_l = \frac{\beta - \rho_l}{\rho_g - \rho_l}, \tag{3.61}$$

$$u_g = \frac{K\rho_l + \frac{(\beta-\rho_l)\omega}{\rho_g-\rho_l}}{\beta}, \quad u_l = \frac{K\rho_g + \frac{(\beta-\rho_g)\omega}{\rho_g-\rho_l}}{\beta}, \quad u_g - u_l = \frac{\omega - K(\rho_g - \rho_l)}{\beta}. \tag{3.62}$$

Thus, α_g, α_l, u_g, and u_l are expressed in terms of β and ω. Using (3.61) and (3.62) in (3.59) and (3.60), and if we assume $\Delta p = \delta p \alpha_g \alpha_l$ with a constant δp , we obtain a system in conservation form for β and ω. By defining the state variable $U = (\beta, \omega)^t$, we can write the system as :

$$U_t + F_x = 0,$$

with flux function :

$$F = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} \alpha_g \rho_l u_g + \alpha_l \rho_g u_l \\ \rho_g \frac{u_g^2}{2} - \rho_l \frac{u_l^2}{2} + \delta p \alpha_g \end{pmatrix}.$$

Remember we assume that $\delta p = \frac{\Delta p}{\alpha_g \alpha_l}$ is a constant. The flux can be written as :

$$F = \begin{pmatrix} \frac{K[\beta(\rho_g+\rho_l)-\rho_g\rho_l]}{\beta} + \frac{(\beta-\rho_g)(\beta-\rho_l)\omega}{\beta(\rho_g-\rho_l)} \\ \frac{(\beta^2-\rho_g\rho_l)\omega^2}{2\beta^2(\rho_g-\rho_l)} + \frac{K\rho_g\rho_l\omega}{\beta^2} - \frac{K^2\rho_g\rho_l(\rho_g-\rho_l)}{2\beta^2} - \frac{\delta p(\beta-\rho_g)}{\rho_g-\rho_l} \end{pmatrix}.$$

From 6.2, we obtain the Jacobian matrix A as follows :

$$A = \begin{pmatrix} \frac{1}{\beta^2}\left[K\rho_g\rho_l + \frac{\omega(\beta^2-\rho_g\rho_l)}{\rho_g-\rho_l}\right] & \frac{(\beta-\rho_g)(\beta-\rho_l)}{\beta(\rho_g-\rho_l)} \\ \frac{[\omega-K(\rho_g-\rho_l)]^2\rho_g\rho_l}{\beta^3(\rho_g-\rho_l)} - \frac{\delta p}{\rho_g-\rho_l} & \frac{1}{\beta^2}\left[K\rho_g\rho_l + \frac{\omega(\beta^2-\rho_g\rho_l)}{\rho_g-\rho_l}\right] \end{pmatrix}.$$

A is a 2×2 matrix with two equal diagonal coefficients. Such a matrix has real eigenvalues provided the extradiagonal terms have the same sign. So the incompressible limit of the two-fluid model is hyperbolic provided :

$$\Delta p \geq \frac{\alpha_g \alpha_l \rho_g \rho_l}{\beta}(u_g - u_l)^2 = \frac{\alpha_g \alpha_l \rho_g \rho_l}{\alpha_g \rho_l + \alpha_l \rho_g}(u_g - u_l)^2.$$

And the eigenvalues and the eigenvectors of the matrix A are :

$$\lambda_{1,2} = \frac{\alpha_g \rho_l u_l + \alpha_l \rho_g u_g}{\beta} \pm \sqrt{\frac{1}{\alpha_g \rho_l + \alpha_l \rho_g}\left(\Delta p - \frac{\alpha_g \alpha_l \rho_g \rho_l}{\alpha_g \rho_l + \alpha_l \rho_g}(u_g - u_l)^2\right)} \tag{3.63}$$

$$= \frac{1}{\beta^2}\left[K\rho_g\rho_l + \frac{\omega(\beta^2 - \rho_g\rho_l)}{\rho_g - \rho_l}\right] \pm \sqrt{\frac{1}{\beta}\left(\Delta p + \frac{(\beta-\rho_g)(\beta-\rho_l)\rho_g\rho_l}{\beta^3}\left(\frac{\omega}{\rho_g-\rho_l} - K\right)^2\right)},$$

$$r_1 = \begin{pmatrix} -\frac{\alpha_g\alpha_l(\rho_g-\rho_l)}{\beta} \\ \sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}(u_g-u_l)^2\right)} \end{pmatrix}, \tag{3.64}$$

$$r_2 = \begin{pmatrix} \frac{\alpha_g\alpha_l(\rho_g-\rho_l)}{\beta} \\ \sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}(u_g-u_l)^2\right)} \end{pmatrix}. \tag{3.65}$$

Then, we can see that the eigenvalues may easily change sign. Now we check if they are either genuinely nonlinear or linearly degenerate.
From Sec. 6.3 we have

$$\begin{aligned}
\nabla\lambda_1 \cdot r_1 &= -\frac{3\alpha_g\alpha_l\rho_g\rho_l(u_g-u_l)}{\beta^3} + \frac{\frac{\Delta p\alpha_g\alpha_l(\rho_g-\rho_l)}{\beta^3} + \frac{(u_g-u_l)^2\rho_g\rho_l\alpha_g\alpha_l\left(\alpha_g\rho_l-\alpha_l\rho_g-2\alpha_g\alpha_l(\rho_g-\rho_l)\right)}{2\beta^4}}{2\sqrt{\frac{1}{\beta}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\beta}(u_g-u_l)^2\right)}} \\
&+ \frac{\alpha_l^2\rho_g - \alpha_g^2\rho_l}{\beta^2}\sqrt{\frac{1}{\beta}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\beta}(u_g-u_l)^2\right)}, \\
\nabla\lambda_2 \cdot r_2 &= \frac{3\alpha_g\alpha_l\rho_g\rho_l(u_g-u_l)}{\beta^3} + \frac{\frac{\Delta p\alpha_g\alpha_l(\rho_g-\rho_l)}{\beta^3} + \frac{(u_g-u_l)^2\rho_g\rho_l\alpha_g\alpha_l\left(\alpha_g\rho_l-\alpha_l\rho_g-2\alpha_g\alpha_l(\rho_g-\rho_l)\right)}{2\beta^4}}{2\sqrt{\frac{1}{\beta}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\beta}(u_g-u_l)^2\right)}} \\
&+ \frac{\alpha_l^2\rho_g - \alpha_g^2\rho_l}{\beta^2}\sqrt{\frac{1}{\beta}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\beta}(u_g-u_l)^2\right)}.
\end{aligned}$$

Conclusion : The field is neither linearly degenerate nor genuinely nonlinear.

3.2.6 Numerical Results

Shock Tube

To validate our scheme, we first use the method to solve a Riemann problem with a pure phase solution.

Problem description This test consists of a Riemann problem with the initial data $\alpha_L = \alpha_R = 0.5$, $p_L = p_R = 1.e5$, $u_{1L} = -u_{1R} = 1$, $u_{2L} = -u_{2R} = -1$, $P_L = P_R = 10^5 Pa$ as in Fig. (3.29, 3.30, 3.31 and 3.32).

In order to assess the influence of the interfacial pressure term on the solutions of the two-fluid model, we compare the numerical solutions obtained with or without the interfacial pressure. We see that, after 168 time steps, the solution obtained without the interfacial pressure blows up. Figs. (3.33, 3.34,3.35 and 3.36) present the solution obtained using upwind method for a mesh of 100 cells, without the interfacial pressure, before blowing up of the calculation. We see that, because of the lack of the hyperbolicity, the pressure and gas velocity increase excessly and we do not capture the correct solution.
We then perform the same test but with the interfacial pressure and obtain the solution as presented in Figs. (3.37 , 3.38, 3.39 and 3.40). We see that, thanks to the interfacial pressure, the pressure does not blow up at time step 168 as in the case without the interfacial pressure. Unfortunately, the gas velocity starts increasing and after some time steps, it blows up. We suppose it is because the Roe's approximate Riemann solver has default of entropy violation. We then apply the entropy fix (presented in the previous section), and we obtain the final solution as presented in Figs.(3.41 , 3.42, 3.43 and 3.44).
It is interesting here to note that, even when the gas vanishes at the center of the computational domain, we can capture its bounded velocity.

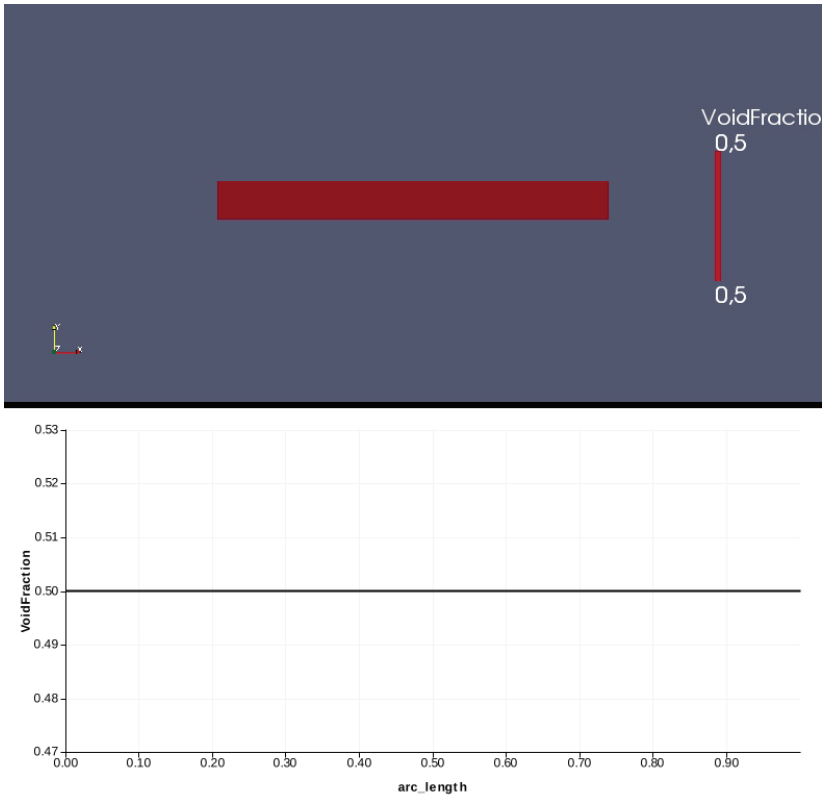

FIGURE 3.29 – *Void fraction initial state*

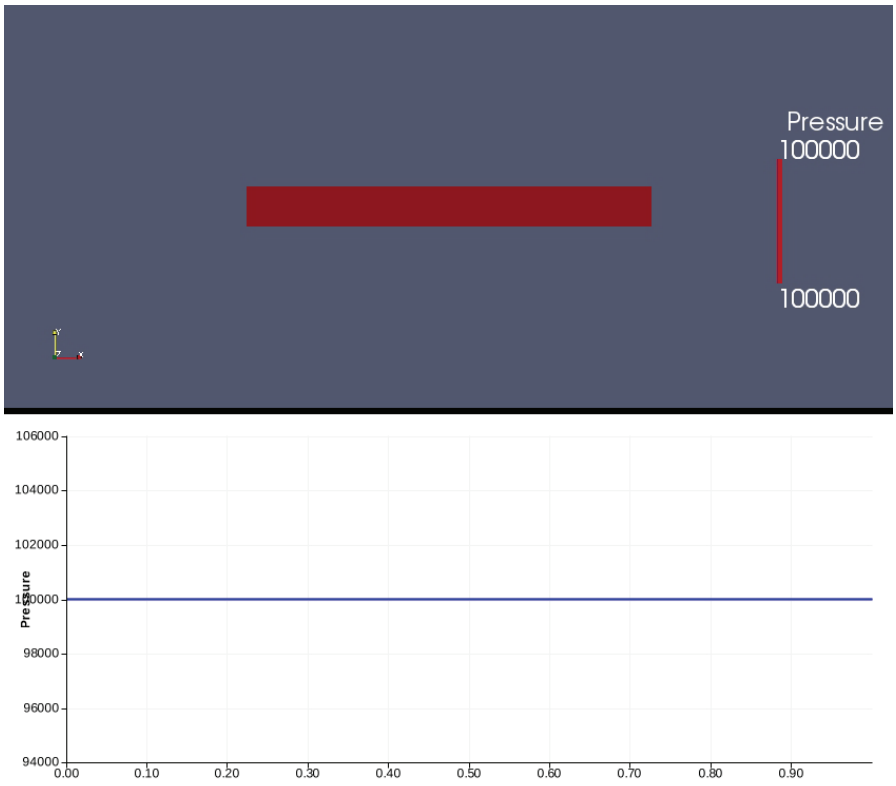

FIGURE 3.30 – *Pressure initial state*

Ransom Faucet Problem

The Ransom faucet problem is described in Ransom (1987), and it has become a common test case for one-dimensional two-fluid models.

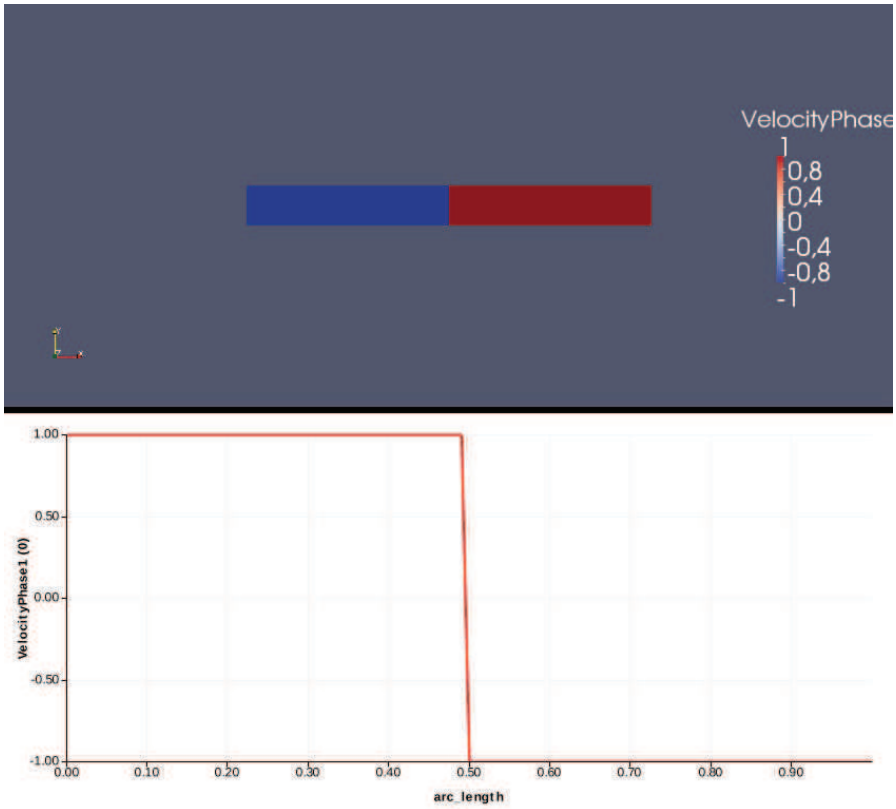

FIGURE 3.31 – *Gas velocity initial state*

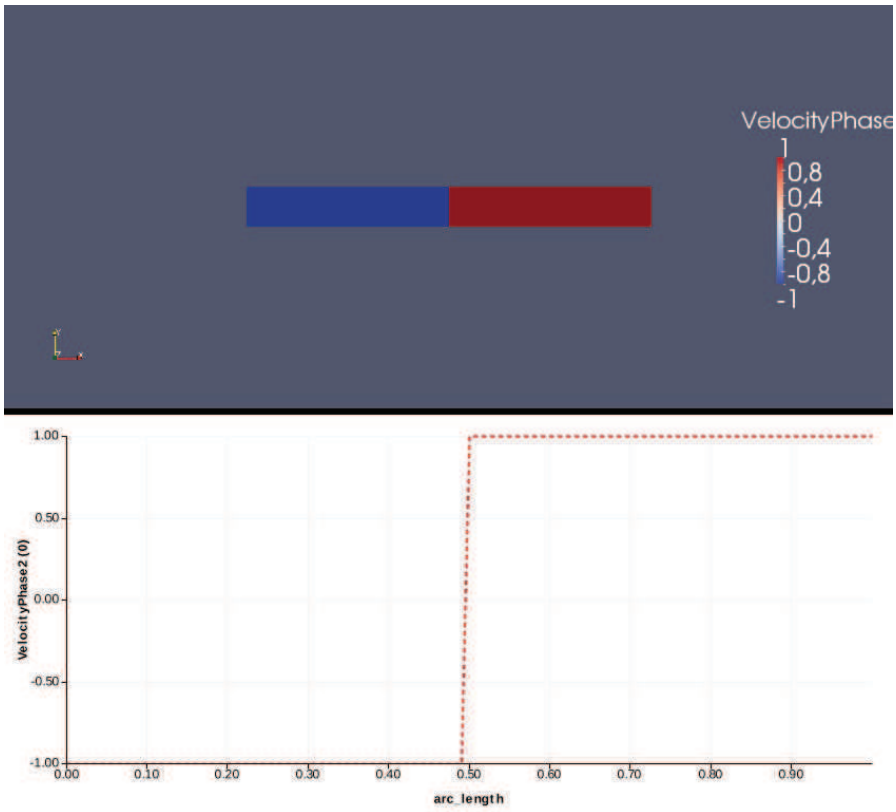

FIGURE 3.32 – *Liquid velocity initial state*

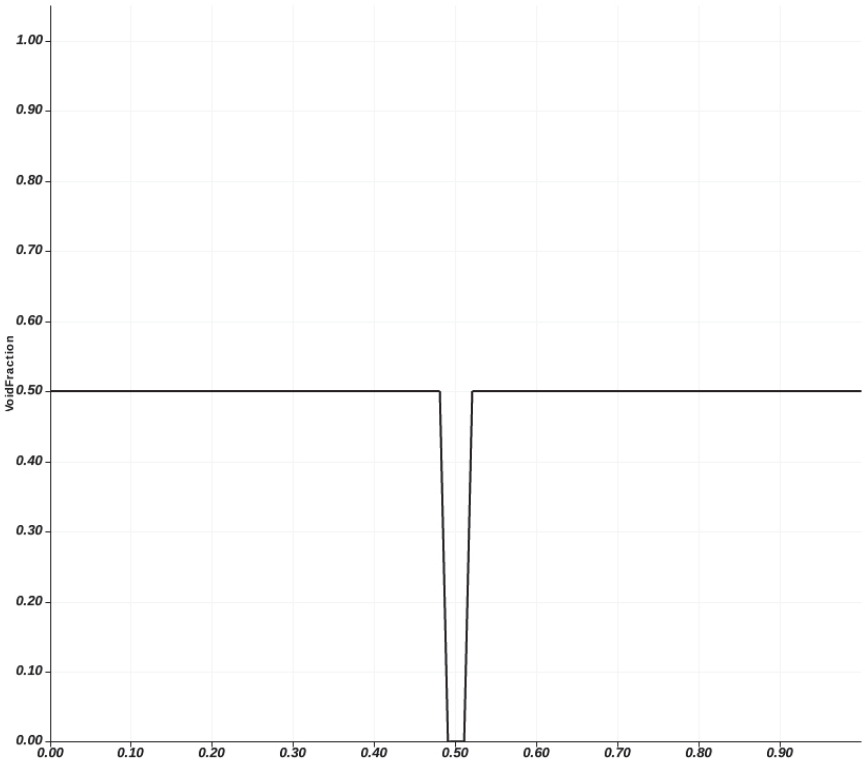

FIGURE 3.33 – *Void fraction, before blowing up*

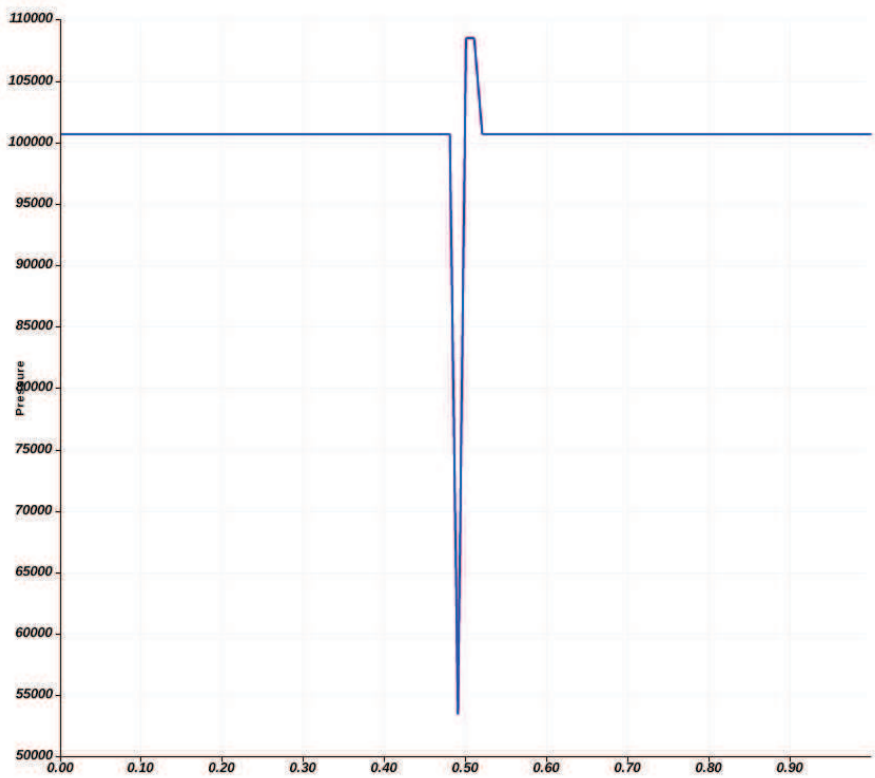

FIGURE 3.34 – *Pressure, before blowing up*

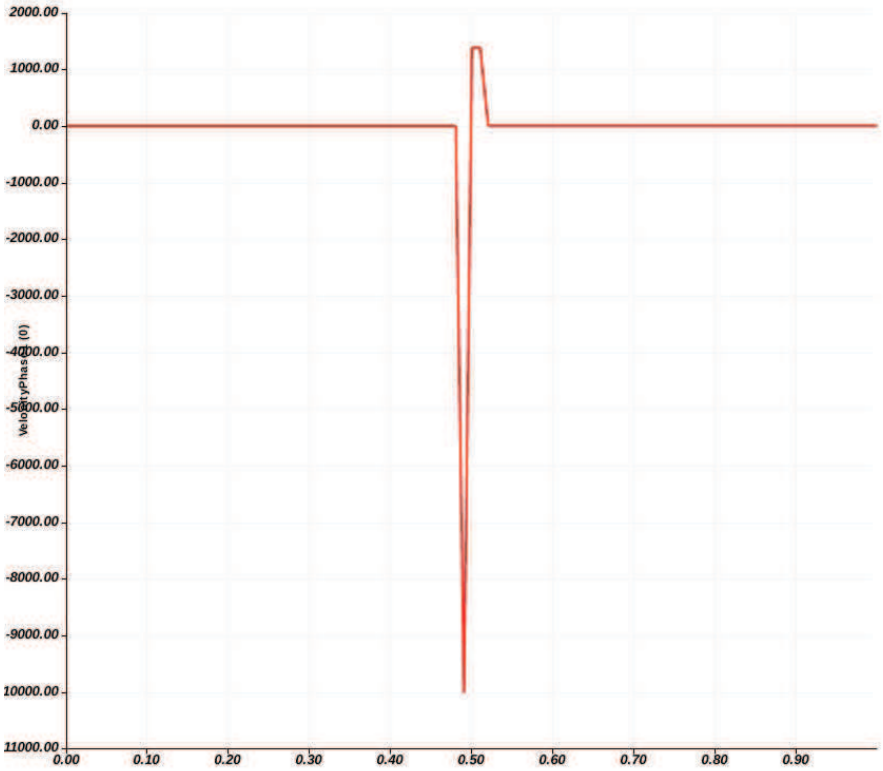

FIGURE 3.35 – *Gas velocity, before blowing up*

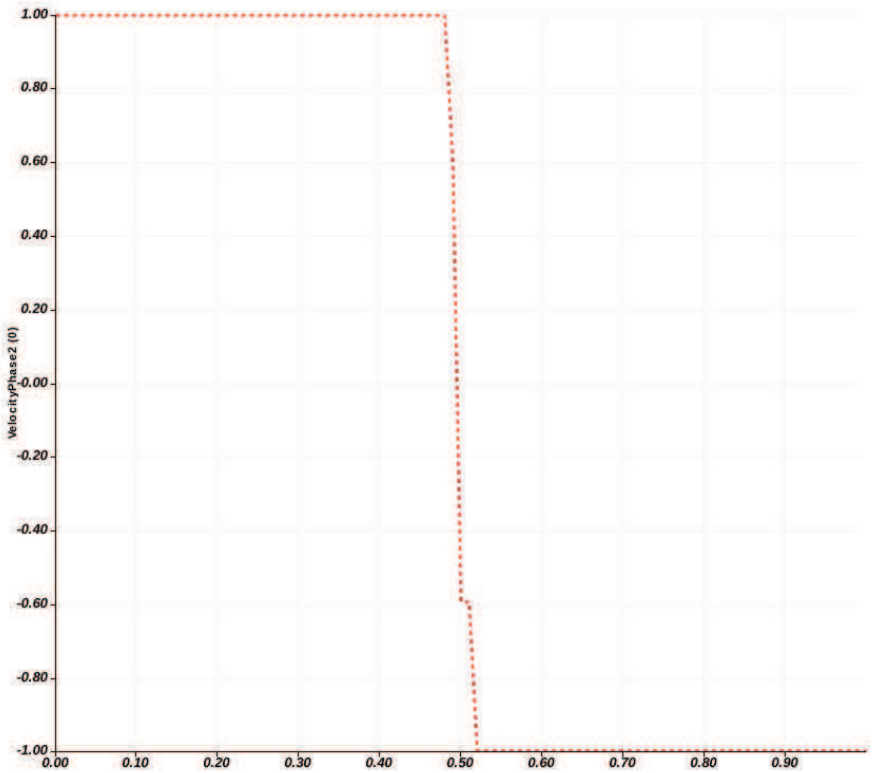

FIGURE 3.36 – *Liquid velocity, before blowing up*

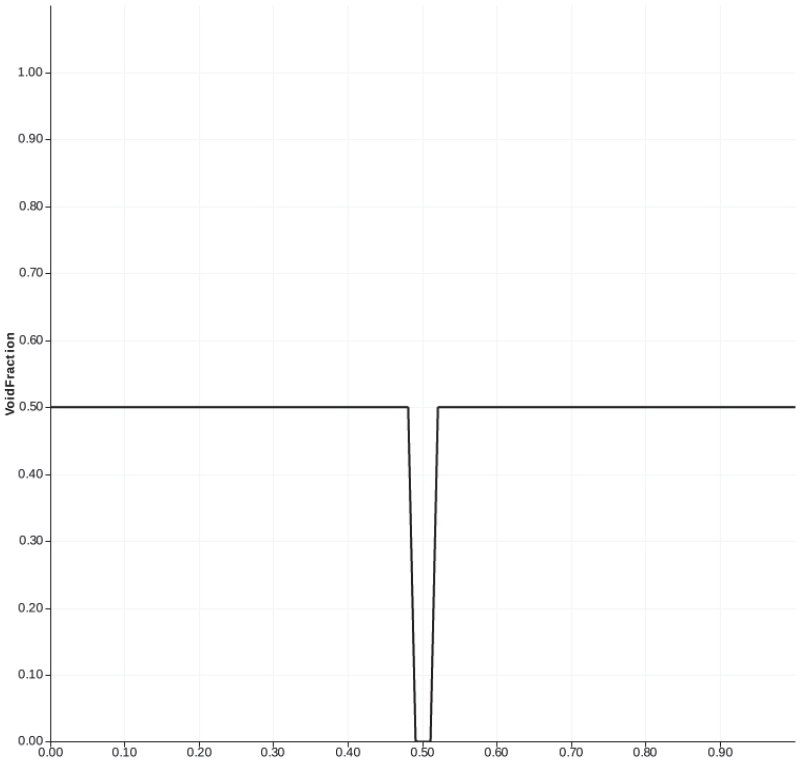

FIGURE 3.37 – *Void fraction, time step* 168

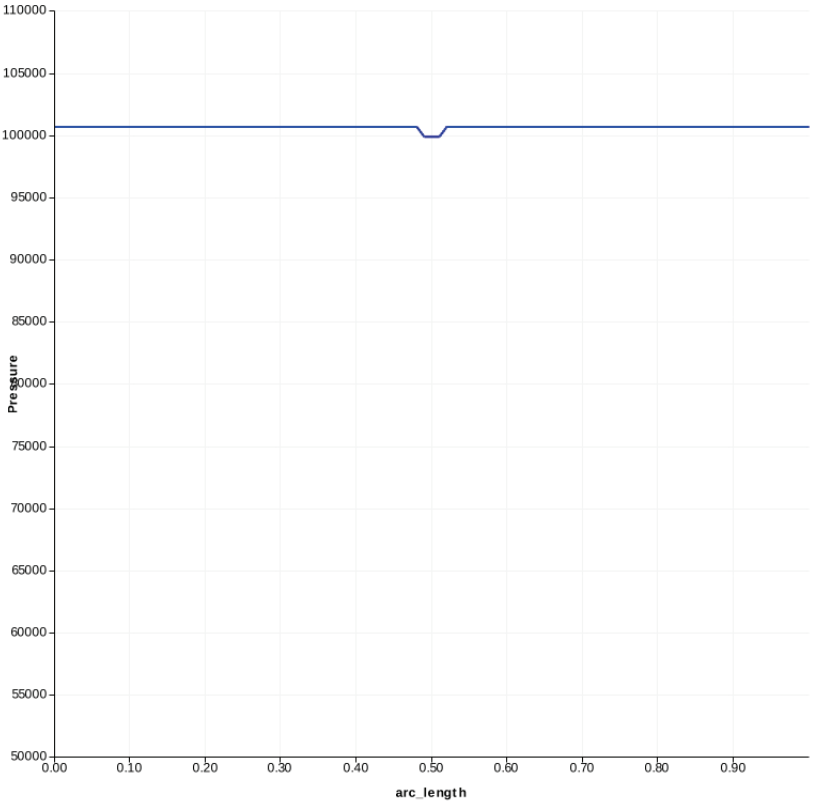

FIGURE 3.38 – *Pressure, time step* 168

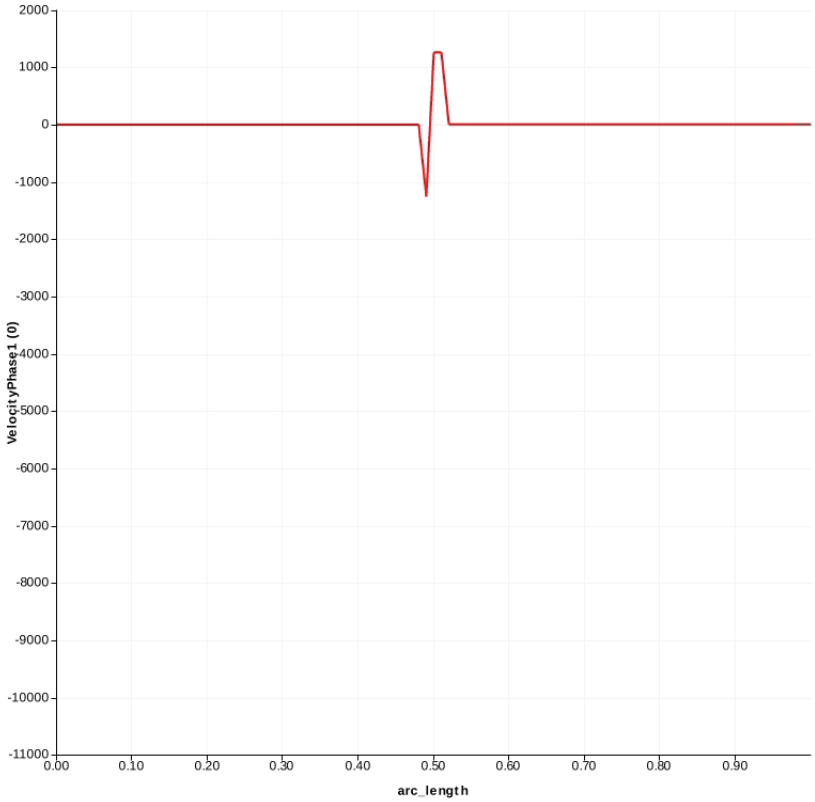

FIGURE 3.39 – *Gas velocity, time step* 168

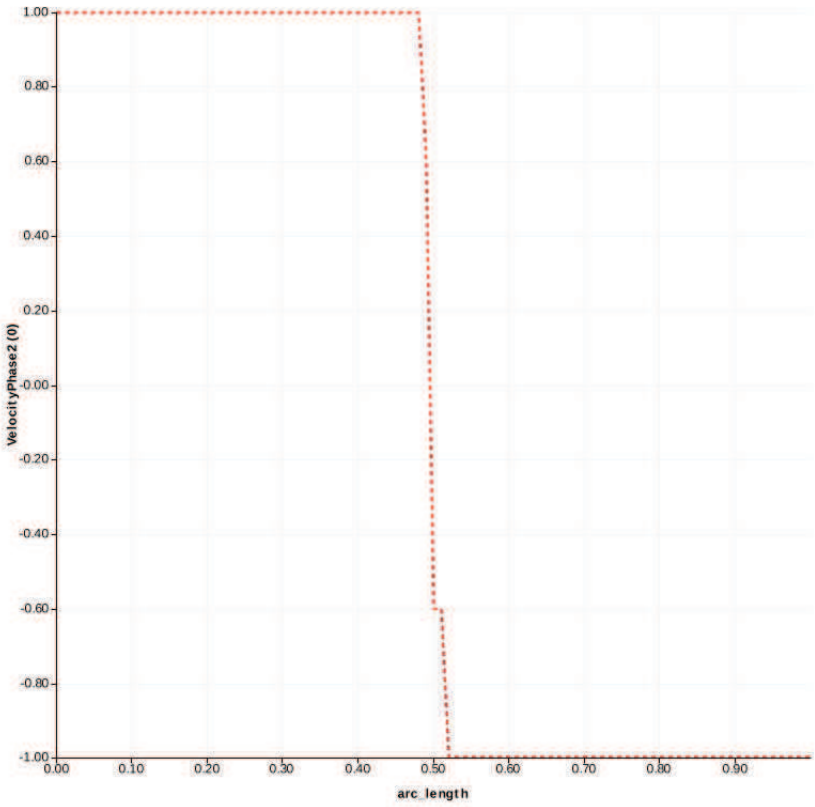

FIGURE 3.40 – *Liquid velocity, time step* 168

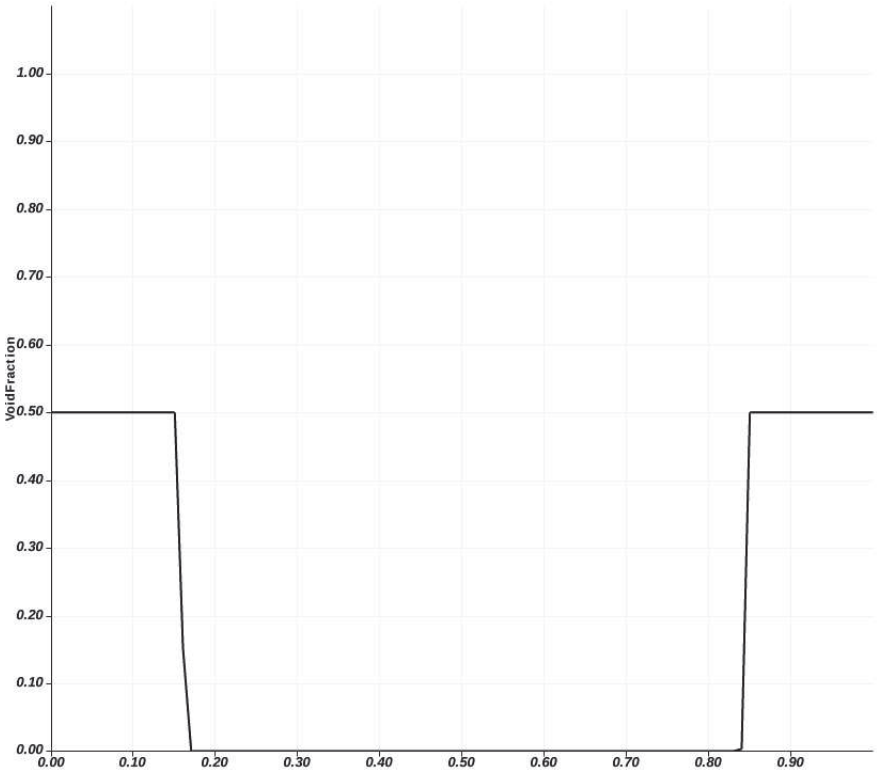

FIGURE 3.41 – *Void fraction*

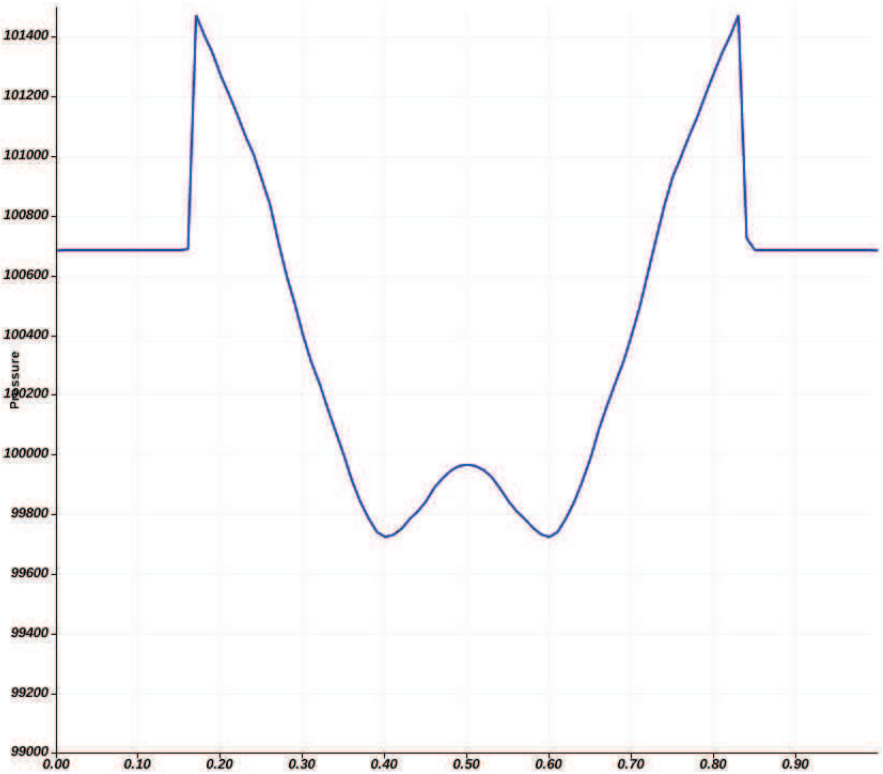

FIGURE 3.42 – *Pressure*

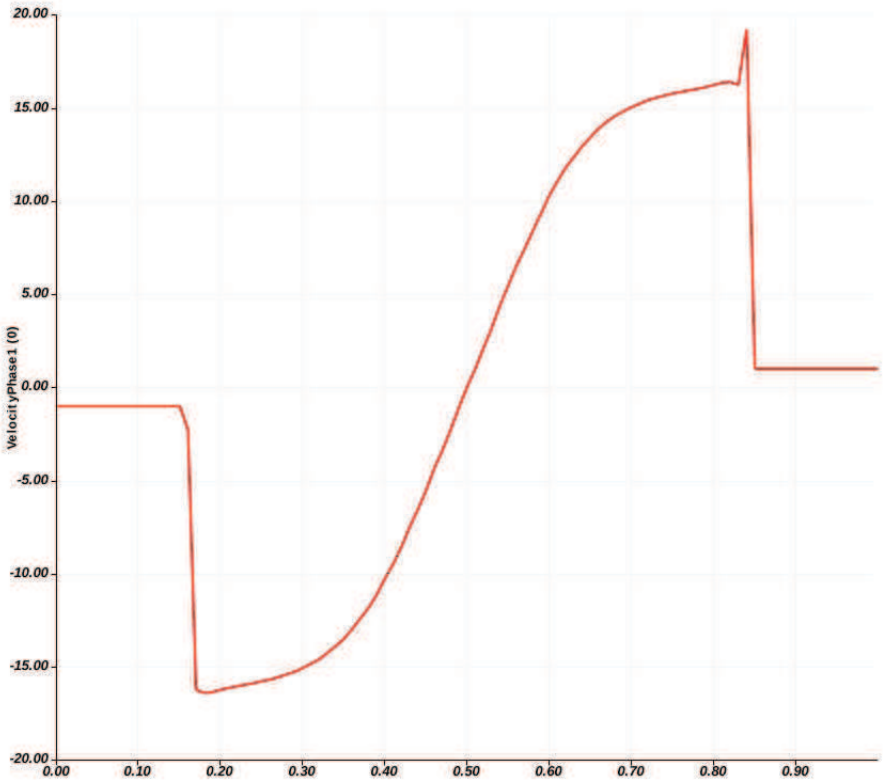

FIGURE 3.43 – *Gas velocity*

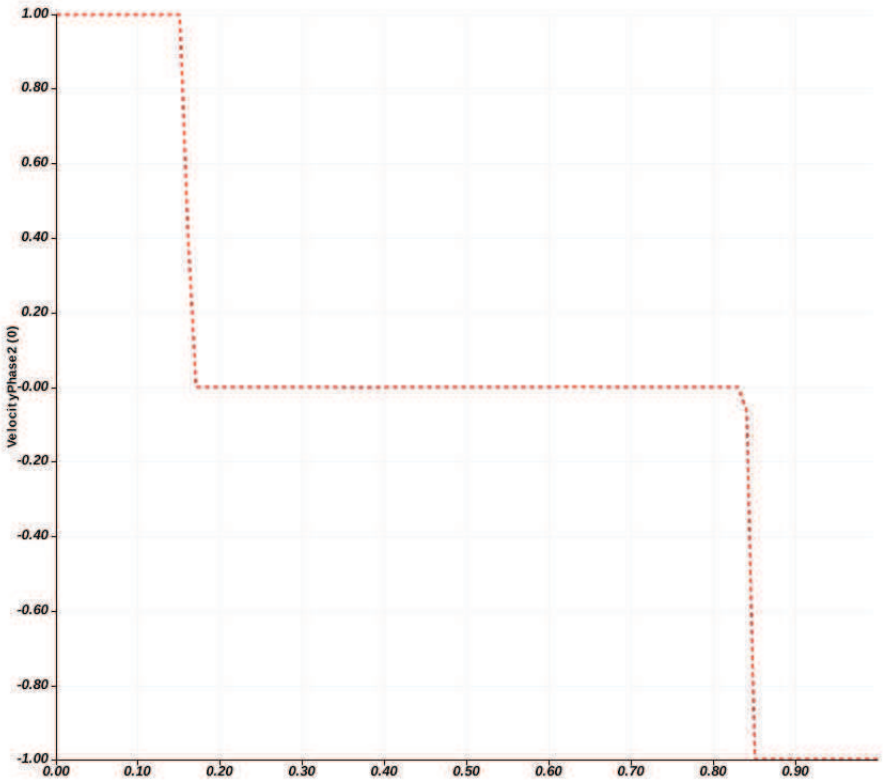

FIGURE 3.44 – *Liquid velocity*

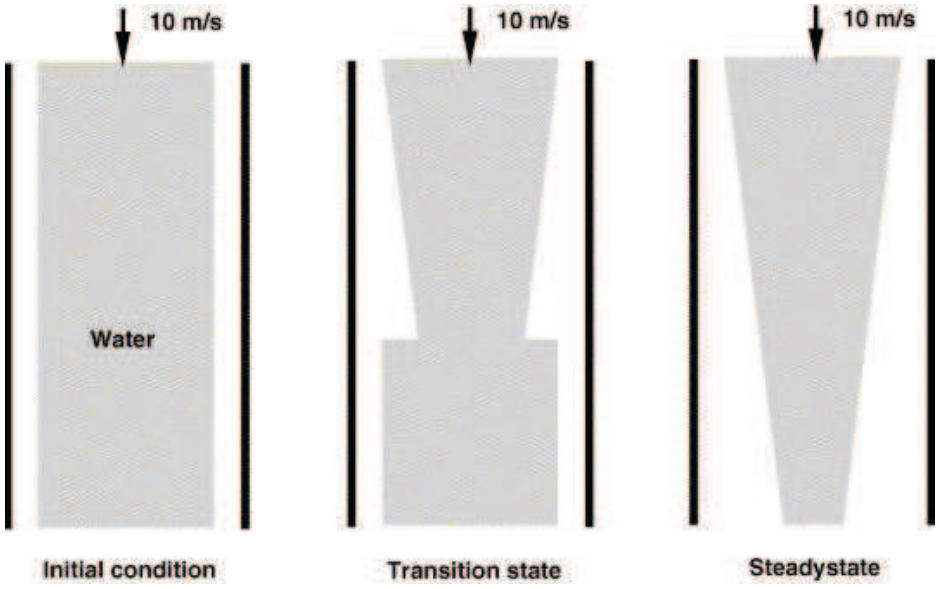

FIGURE 3.45 – *Illustration of Ransom's water faucet problem*

Problem description Referring to Fig.3.45, the test consists of a vertical tube 12 m in length and 1 m in diameter. At the initial state, the liquid, surrounded by stagnant gas, has a uniform velocity of $u_l^0 = 10$ m/s, the gas volume fraction is $\alpha_g^0 = 0.2$, the pressure is equal to $10^5 Pa$, and the temperature is equal to 50^0 C. Then the gravity force is applied to the fluid, causing the water column to accelerate and get narrower at the bottom. The boundary conditions at the inlet are specified velocities of $u_l = 10$ m/s for the liquid and of $u_g = 0$ for the gas (vapor), of $\alpha_g = 0.2$ for vapor volume fraction, of 50^0 C for the temperature, and pressure is extrapolated from the computational domain. At the outlet, the pressure is imposed equal to $10^5 Pa$, while all other quantities are extrapolated from the interior domain.
This problem admits an analytical solution, derived under the assumptions that the liquid is incompressible and that variations of pressure in the liquid may be neglected. Coquel et al. (1997) provided the solution for the gas volume fraction profile :

$$\alpha_g(x,t) = \begin{cases} 1 - \frac{(1-\alpha_g)u_l^0}{\sqrt{2gx+(u_l^0)^2}} & \text{if } x \leq u_l^0 t + \frac{1}{2}gt^2, \\ \alpha_g^0 & \text{otherwise,} \end{cases}$$

and the expression for the liquid and velocity is given by

$$u_l(x,t) = \begin{cases} \sqrt{2gx + (u_l^0)^2} & \text{if } x \leq u_l^0 t + \frac{1}{2}gt^2, \\ u_l^0 + gt & \text{otherwise,} \end{cases}$$

Using the explicit upwind scheme, we run the test on numerical grids ranging from 101, 201, 1001 grid points. The CFL number is 1. First, the gas volume fraction profiles are presented at time $t = 0.6$s in Fig.3.46. It is seen that as the mesh is refined the numerical solution converges to the exact one, but it is a little slow. In fact, the convergence was less than first order, even if the scheme is formally first-order accurate in space. This is perharp due to the discontinuity in the solution. For discontinuous

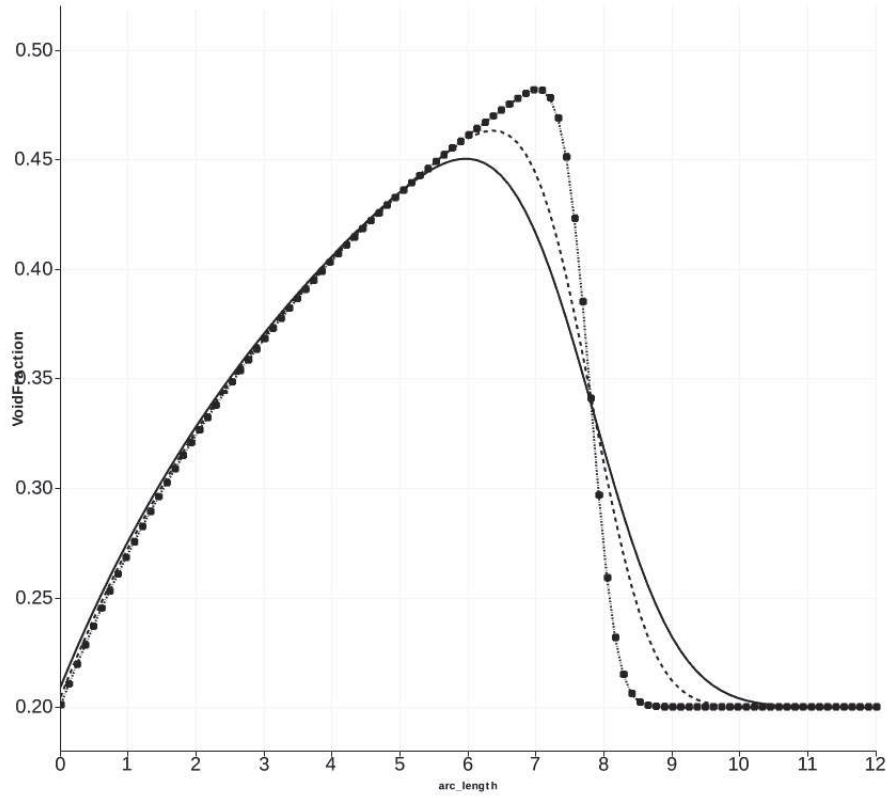

FIGURE 3.46 – *Gas volume fraction for the water faucet,* 101, 201, 1001 *grid points*

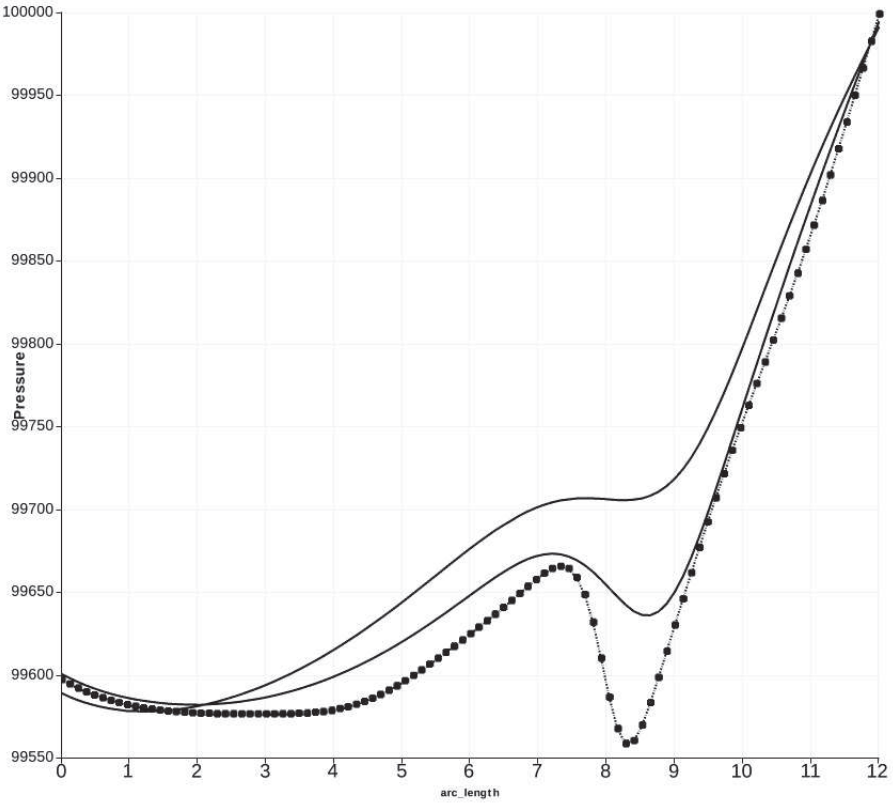

FIGURE 3.47 – *Pressure for the water faucet,* 101, 201, 1001 *grid points*

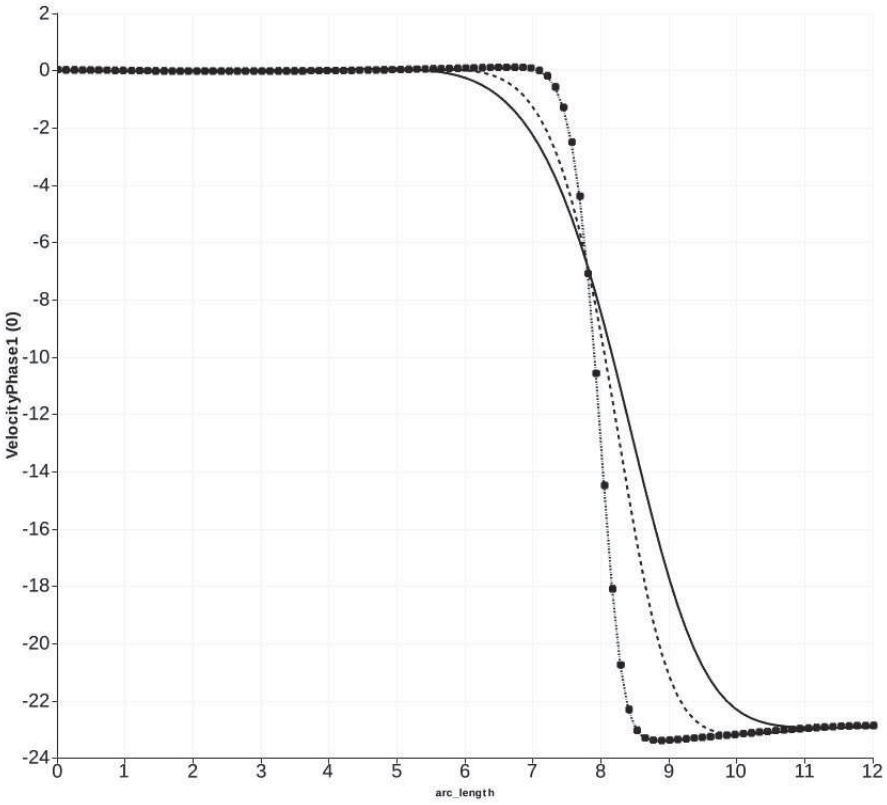

FIGURE 3.48 – *Gas velocity for the water faucet*, 101, 201, 1001 *grid points*

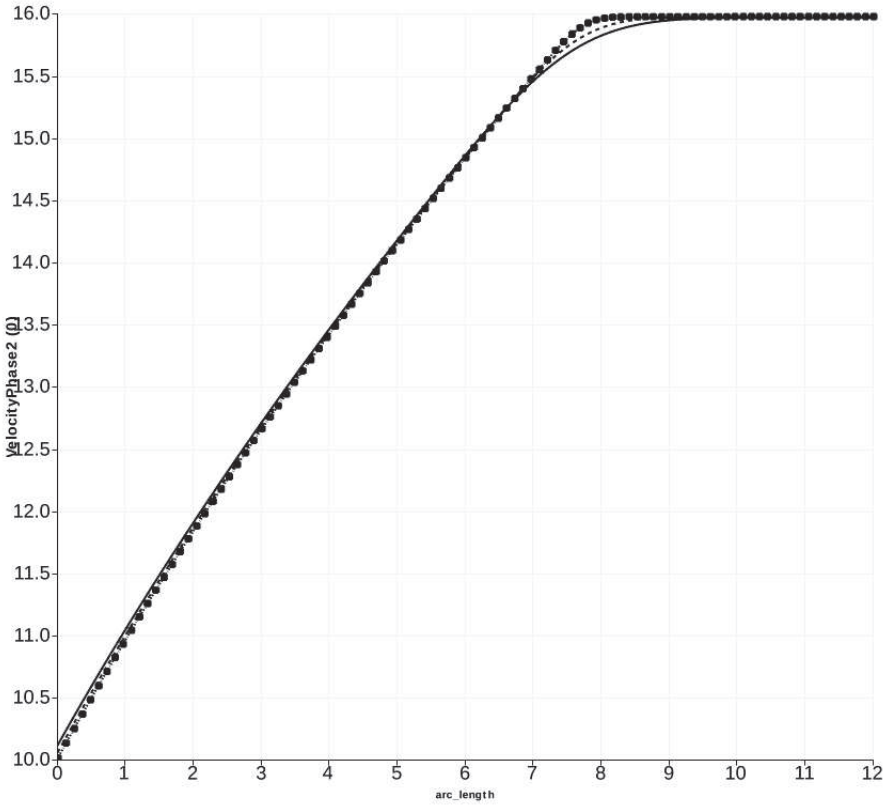

FIGURE 3.49 – *Liquid velocity for the water faucet*, 101, 201, 1001 *grid points*

solutions, the smooth solution order of the scheme can normally not be attained (see LeVeque (2002), Section 8.7).

The pressure is by far the most sensitive variable in the faucet case. Fig. (3.47) presents the pressure profile in our test case. The remaining physical variables, namely the gas and liquid velocities are displayed in Figs. (3.48 and 3.49). It should be noted here that, in the case of water faucet, both schemes (with or without entropy fix) give the correct solution.

Sedimentation Problem

This problem is a simplified gravity-induced phase separation problem proposed by Coquel et al. (1997). The most important characteristic of the sedimentation problem is the transition from two-phase flow to single-phase flow.

Problem description The sedimentation problem consists of a vertical pipe of length $L = 7.5$m, closed at both ends. Like in the case of the water faucet, gravity is the only source term considered. Initially it has a uniform pressure of $P = 10^5$ Pa and a volume fraction of $\alpha_g = \alpha_l = 0.5$. Velocities at the top and the bottom boundaries are considered to be zero. The gravity force provides a separation of the phases for $t > 0$. The solution at $t = \infty$ is composed of a pure gas at rest at the top side, and a liquid at rest at the bottom side of the pipe. A schematic representation of the case is depicted in Figs.3.50 and 3.51.

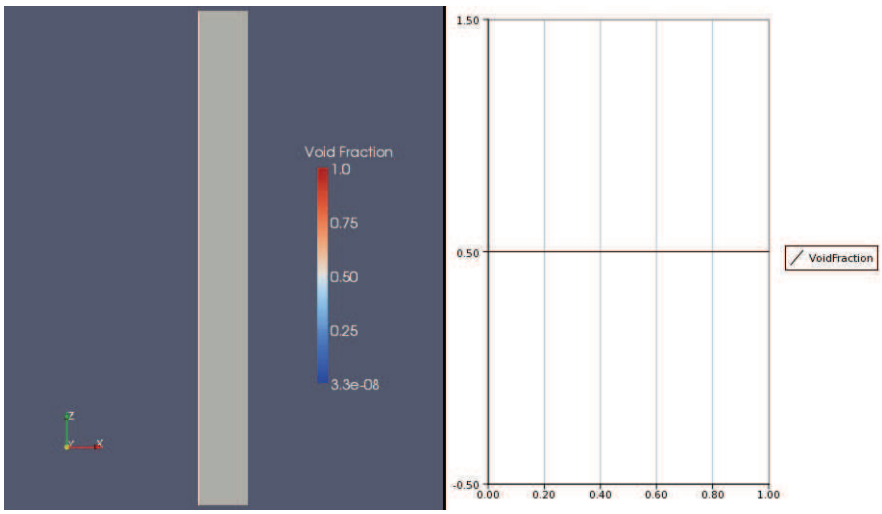

FIGURE 3.50 – *Initial state*

In fact, Evje and Flatten (2003) derived approximate analytical solutions for the liquid velocity and volume fraction by assuming that the liquid is accelerated by gravity only, until it is abruptly brought into stag-

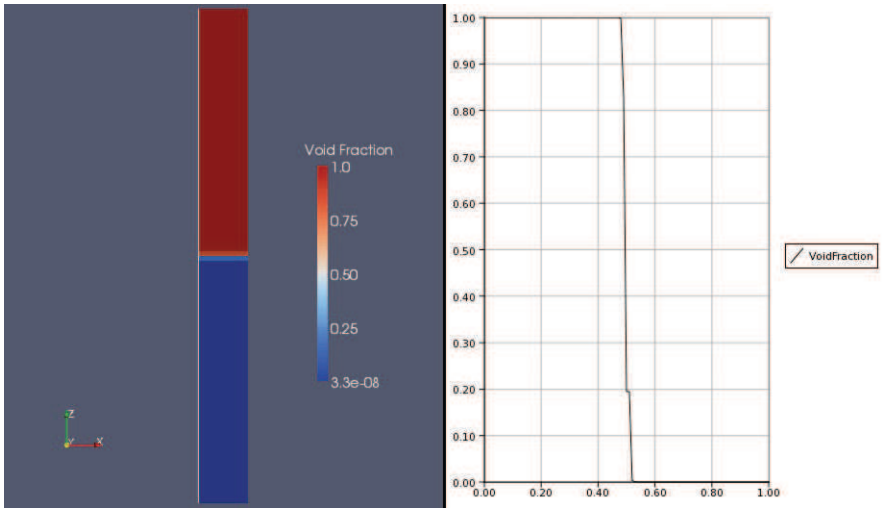

FIGURE 3.51 – *Final state*

nation at the lower part of the tube :

$$\alpha_l(x,t) = \begin{cases} 0 & \text{for } x \leq \frac{1}{2}gt^2, \\ 0.5 & \text{for } \frac{1}{2}gt^2 \leq x \leq L - \frac{1}{2}gt^2, \\ 0 & \text{for } L - \frac{1}{2}gt^2 \leq x, \end{cases}$$

and

$$u_l(x,t) = \begin{cases} \sqrt{2gx} & \text{for } x \leq \frac{1}{2}gt^2, \\ gt & \text{for } \frac{1}{2}gt^2 \leq x \leq L - \frac{1}{2}gt^2, \\ 0 & \text{for } L - \frac{1}{2}gt^2 \leq x, \end{cases}$$

After the time

$$t = \sqrt{\frac{L}{g}} \approx 0.87s,$$

the phases should be fully separated in the idealized case. Numerical results have been compared with the previous analytical solution. In the following, we present the numerical solution obtained using an upwind scheme with entropy fix on a mesh of 100 cells, CFL = 1. It should be noted here that only with using entropy fix, we can solve the problem. If we do not use the entropy fix, the Roe's approximate Riemann solver does not work. The computation breaks down because of the negative phasic mass.

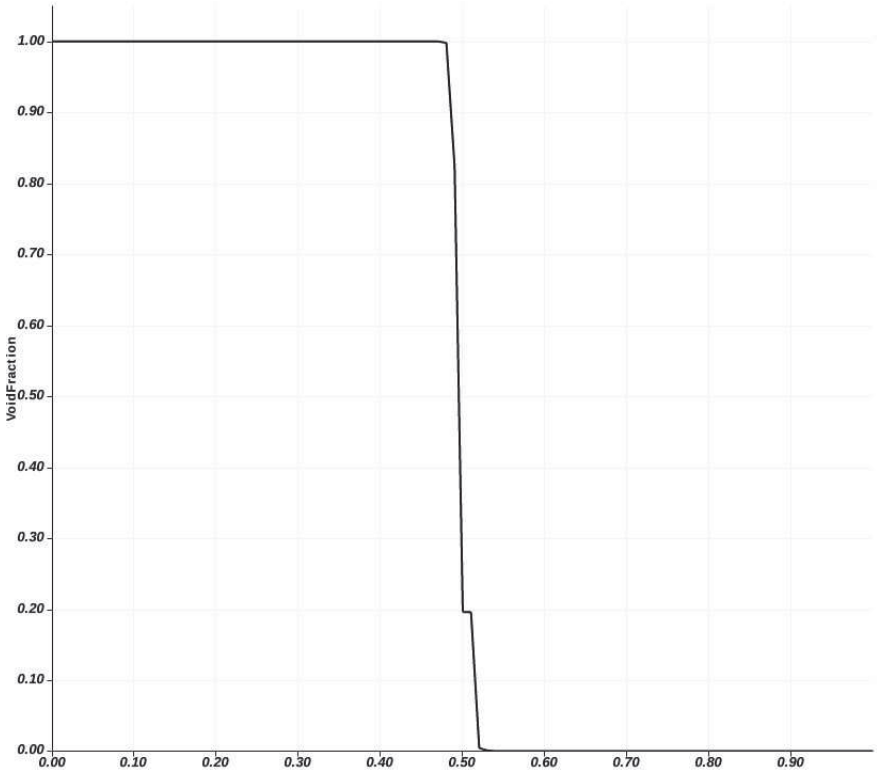

FIGURE 3.52 – *Void fraction, sedimentation*

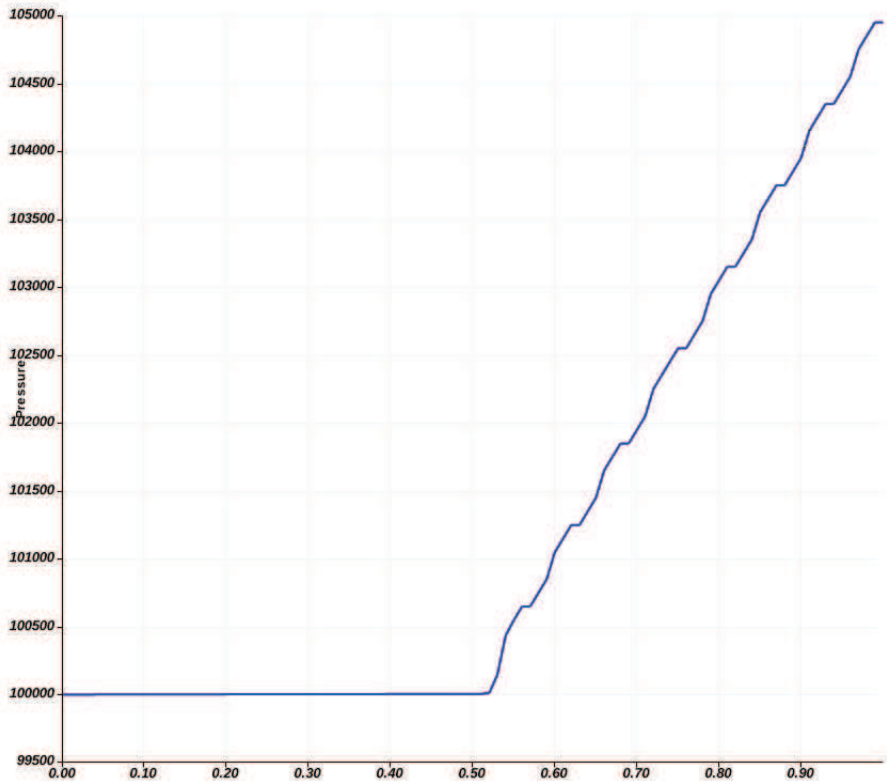

FIGURE 3.53 – *Pressure, sedimentation*

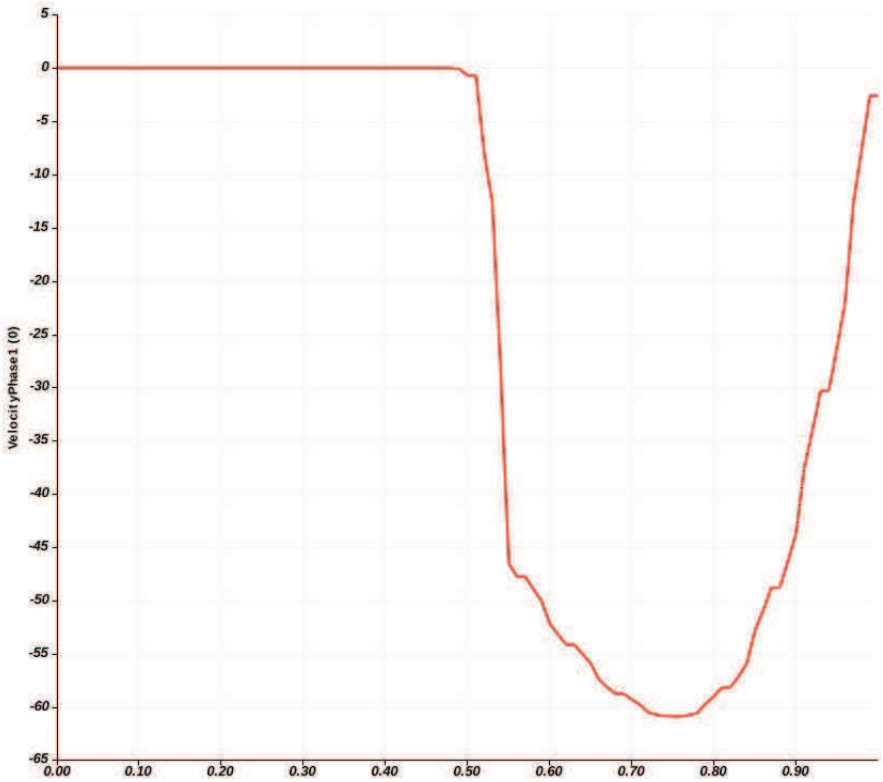

FIGURE 3.54 – *Gas velocity, sedimentation*

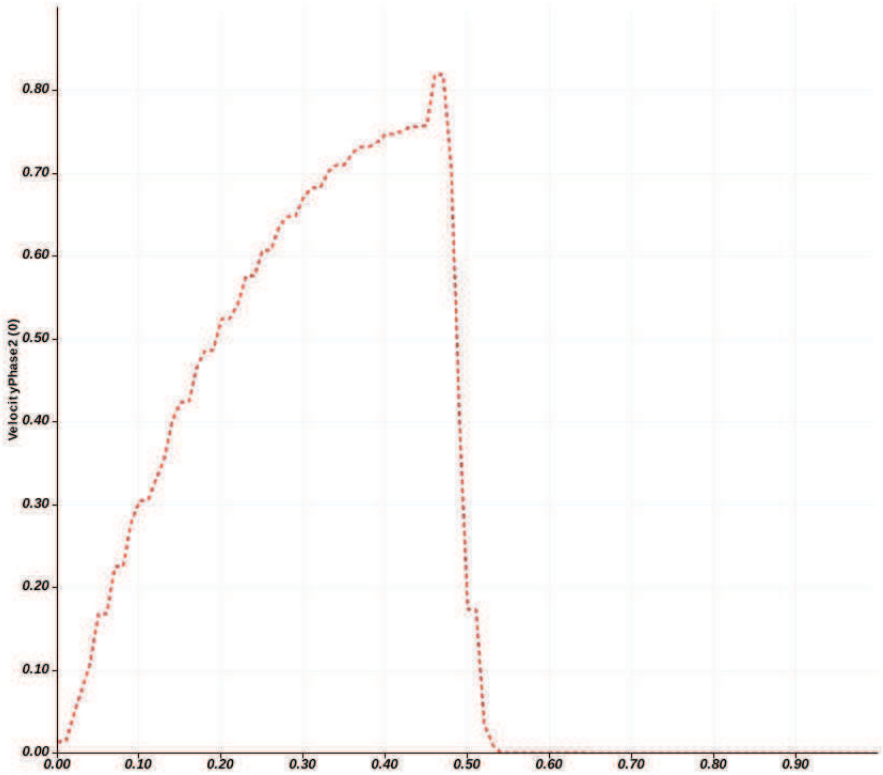

FIGURE 3.55 – *Liquid velocity, sedimentation*

Domain Decomposition Method

4

Contents

In the numerical resolution of partial differential equations, we generally have to solve linear or nonlinear systems arising from the discretization. The large size and the fact these systems are ill conditionned make a global resolution difficult. In fact, direct solvers are too costly and iterative solvers are not robust enough. There is a need for hybrid iterative/direct solvers : these are domain decomposition methods. Roughly speaking, the computational domain is decomposed into smaller subdomains. The problems on the subdomains are independent and can be solved by a direct or iterative method and the matching of the solutions is imposed iteratively. Furthermore, with the advent of parallel computers, domain decomposition methods have enjoyed an increasing popularity among the scientific community because they are well adapted to parallel computers. Indeed, since early 1980's, there has been gratifying progress in the development of domain decomposition algorithms for symmetric elliptic problems, and a number of fast methods have been designed for which the condition number of the matrix is uniformly bounded or grows only in proportion to $1 + ln(H/h)$ where H is the diameter of a typical subdomain and h is the diameter of a typical element into which the subdomains are divided. Such algorithms are often called optimal or nearly optimal algorithms, respectively, though we note that these adjectives pertain to the convergence rate only, and not to the overall computational

complexity. For hyperbolic or mixed hyperbolic parabolic problems of compressible fluid mechanics, the theory to date is far less satisfactory. For such problems, one has to deal with first-order PDEs characterized by nonsymmetric operators, with possible singular solutions. The solution of such problems is often CPU-bound or memory-bound or both on the fastest and largest computers available.

Nowadays, for linear problems, domain decomposition methods can be viewed as preconditioners for Krylov iterative methods such as the conjugate gradient (CG) method or the method of generalized minimum residual (GMRES). For nonlinear problems, they may be viewed as preconditioners for the solution of the linear systems arising from the use of Newton's method or as preconditioners for solvers such as the nonlinear conjugate gradient method.

As pointed in Smith et al. (1996), the term domain decomposition covers a fairly large range of computing techniques for the numerical solution of partial differential equations (PDEs) in time and space and has slightly different meanings to specialists within the discipline of PDEs.

- In parallel computing, it means the process of distributing data from a computational model among the processors in a distributed memory computer.
- In asymptotic analysis, it means the separation of the physical domain into regions which can be modeled with different equations.
- In preconditioning methods, it refers to the process of splitting the global problem into smaller ones that are easily solved and whose solutions can be used to produce a preconditioner for the original operator.

Note that all three of these may occur in a single program.

Three of the more important motivations for domain decomposition methods are

- ease of parallelization and good parallel performance,
- simplification of problems on complicated geometry, and
- superior convergence properties.

Domain decomposition methods are commonly classified according to a few orthogonal criteria."Overlapping" and "nonoverlapping" methods are differentiated by the decomposition into territories on which the elemental subproblems are defined. Overlapping methods generally permit simple Dirichlet updating of the boundary data of the subdomains at the expense of extra arithmetic complexity per iteration from the redundantly defined degrees of freedom. "Additive" (Jacobi-like) or "multiplicative" (Gauss-Seidel-like) methods are differentiated by the interdependence of the subdomains within each iteration. For the same number of subdomains additive methods are intrinsically more parallelizable.

In this chapter, we first give an overview of some popular domain decomposition methods for finite element discretizations. We then introduce our demain decomposition methods using finite volume discretizations.

4.1 Review on Domain Decomposition Methods for Finite Element Methods

In this section, we will use an algebraic approach to give an overview of some popular domain decomposition methods.

Consider a finite element mesh covering the computational domain Ω. After the discretization, we obtain a linear system $Ax = b$ where A is a sparse matrix. To describe domain decomposition methods using an algebraic approach, as in Cai and Saad (1996), we define a connectivity graph $G_\Omega = (W_\Omega, E_\Omega)$ where the set of graph vertices $W_\Omega = \{1, \cdots, n_e\}$ represents the n_e elements in the finite element mesh and the edges set $E_\Omega = \{(i,j)\, s.t\, a_{ij} \neq 0\}$ represents the pairs of vertices that are coupled by a nonzero element in A. Assume that the connectivity graph has been partitioned resulting in N nonoverlapping subsets Ω_i^0 whose union is Ω. These subsets are referred to as subdomains and are also often referred to as substructures. The Ω_i^0 can be generalized to overlapping subsets of graph vertices. In particular, construct Ω_i^1, the one-overlap decomposition of Ω, by taking Ω_i^0 and including all graph vertices corresponding to immediate neighbours of the vertices in Ω_i^0. By recursively applying this definition, the δ-layer overlap ($\delta \in \mathbb{N}$) of W_Ω is constructed and the subdomains are denoted Ω_i^δ.
Corresponding to each subdomain Ω_i^0 we define a rectangular extension matrix $R_i^{0^T}$ whose action extends by zero a vector of values defined at mesh vertices associated with the finite elements contained in Ω_i^0. The entries of $R_i^{0^T}$ are zeros and ones. For simplicity, we omit the 0 superscripts and define $R_i = R_i^0$ and $\Omega_i = \Omega_i^0$. Note that the columns of a given R_i are orthogonal, but that between the different R_i's some columns are no longer orthogonal. This is due to the fact that some mesh vertives overlap even though the graph vertices defined by Ω_i are nonoverlapping. Let Γ_i be the set of all mesh vertives belonging to the interface of Ω_i ; that is mesh vertices lying on $\partial\Omega_i \backslash \partial\Omega$. Similarly, let I_i be the set of all remaining mesh vertices within the subdomain Ω_i (i.e. interior vertices). Considering only the discrete matrix contributions arising from finite elements in Ω_i gives rise to the following local discretization matrix :

$$A_i = \begin{pmatrix} A_{I_i I_i} & A_{I_i \Gamma_i} \\ A_{\Gamma_i I_i} & A_{\Gamma_i \Gamma_i} \end{pmatrix}, \tag{4.1}$$

where interior vertices have been ordered first. The matrix A_i corresponds to the discretization of the PDE on Ω_i and the one-one block $A_{I_i I_i}$ corresponds to the discretization with homogeneous Dirichlet conditions on Γ_i. The completely assembled discretization matrix is obtained by summing the contributions over the substructures/subdomains :

$$A = \sum_{i=1}^{N} R_i^T A_i R_i. \tag{4.2}$$

In a parallel distributed environment each subdomain is assigned to one processor and typically processor i stores A_i. A matrix-vector product is performed in two steps. First a local matrix-vector product involving A_i

is performed followed by a communication step to assemble the results along the interface Γ_i.
For the $\delta-$overlap partition we can define a corresponding restriction operator R_i^δ which maps mesh vertices in Ω to the subset of mesh vertices associated with finite elements contained in Ω_i^δ. Corresponding definitions of Γ_i^δ and I_i^δ follow naturally as the boundary and interior mesh vertices associated with finite elements in Ω_i^δ. The discretization matrix has a similar structure to the one given by (4.1) and is written as

$$A_i^\delta = \begin{pmatrix} A_{I_i^\delta I_i^\delta} & A_{I_i^\delta \Gamma_i^\delta} \\ A_{\Gamma_i^\delta I_i^\delta} & A_{\Gamma_i^\delta \Gamma_i^\delta} \end{pmatrix}. \tag{4.3}$$

4.1.1 Overlapping and Nonoverlapping Schwarz Methods

The first domain decomposition methods are most often referred to as Schwarz methods due to the pioneering work of Schwarz (1869). This work was not intended as a numerical algorithm but was instead developed to show the existence of the elliptic problem solution on a complex geometry formed by overlapping two simple geometries where solutions are known. With the advent of parallel computing this basic technique, known as the alternating Schwarz method, has motivated considerable research activity. In this section, we do not intend to give an exhaustive presentation of all works devoted to Schwarz methods. Only additive variants that are well-suited to straightforward parallel implementation are considered. Within additive variants, computations on all subdomains are performed simultaneously while multiplicative variants require some subdomain calculations to wait for results from other subdomains. The multiplicative versions often have connections to block Gauss-Seidel methods while the additive variants correspond more closely to block Jacobi methods. We do not further pursue this description but refer the interested reader to Smith et al. (1996) and Toselli and Widlund (2005).

Additive Schwarz preconditioners

With the notations in the previous section, the additive Schwarz preconditioner is given by

$$M_{AS}^\delta = \sum_{i=1}^{N} (R_i^\delta)^T (A_{I_i I_i}^\delta)^{-1} R_i^\delta. \tag{4.4}$$

Here the $\delta-$overlap is defined in terms of finite element decompositions. The preconditioner and the R_i^δ operators, however, act on mesh vertices corresponding to the submeshes associated with the finite element decomposition. The preconditioner is symmetric (respectively symmetric positive definite) if the original system, A, is symmetric (respectively symmetric positive definite).
Parallel implementation of this preconditioner requires a factorization of a Dirichlet problem on each processor in the setup phase. Each invocation of the preconditioner requires two neighbour to neighbour communications. The first corresponds to obtaining values within overlapping regions associated with the restriction operator. The second corresponds to summing

the results of the backward/forward substitution via the extension operator.
In general, larger overlap usually leads to faster convergence up to a certain point where increasing the overlap does not further improve the convergence rate. Unfortunately, larger overlap implies greater communication and computation requirements.

Restricted Additive Schwarz preconditioner

A variant of the classical additive Schwarz method is introduced in Cai and Sarkis (1999) which avoids one communication step when applying the preconditioner. This variant is referred to as Restricted Additive Schwarz (RAS). It does not have a natural counterpart in a mesh partitioning framework that by construction has overlapping sets of vertices. Consequently, the closest mesh partitioning counterpart solves a Dirichlet problem on a large subdomain but considers the solution only within the substructure. That is

$$M^{\delta}_{RAS} = \sum_{i=1}^{N} (R_i)^T (A^{\delta}_{I_i I_i})^{-1} R^{\delta}_i. \tag{4.5}$$

Surprisingly, M^{δ}_{RAS} often converges faster than M^{δ}_{AS} and only requires half the communication making it frequently superior on parallel distributed computers. Of course it might not be suitable for symmetric positive definite problems because M^{δ}_{RAS} is nonsymmetric even for symmetric matrix A.

Multiplicative Schwarz Method

The multiplicative Schwarz algorithm is a direct extension of the classical Schwarz alternating algorithm, introduced in 1969 by H.A. Schwarz in an existence proof for some elliptic boundary value problems in certain irregular regions. This method has attracted much attention as a convenient computational method for the solution of a large class of elliptic or parabolic equations.

Non-overlapping Schwarz method

A variant of the alternating Schwarz algorithm involving independant solutions in each subdomains has been developed by Lions (1989) and Lions (1990) by changing the iterative process, making the algorithm suitable for parallel computing. As mentionned previously, the convergence speed of the Schwarz algorithm is proportional to the size of the overlap between the subdomains. A variant can be formulated with non-overlapping subdomains and the transmission conditions should be changed from Dirichlet to Robin (Després et al. (1992), Gander et al. (2007)). These absorbing boundary transmission conditions defined on the interface between the non-overlapping subdomains, are the key ingredients to obtain a fast convergence of the iterative Schwarz algorithm (Quarteroni

and Valli (1999),Maday and Magoulès (2006)). Despite optimal transmission conditions can be derived, they consists of non local operators and thus are not easy to implement in a parallel computational environment. An alternative consists to approximate these operators with partial differential operators as investigated in (Japhet et al. (2001)) for convection diffusion equations, in (Després et al. (1992), Dolean and Lanteri (2001)) for Maxwell equation, and (Chevalier and Nataf (1998), Gander et al. (2002), Magoulès et al. (2004a), Gander et al. (2007)) for the Helmholtz equation where a minimization procedure has been used. An another alternative consists to approximate the optimal transmission conditions in an algebraic way, as investigated in (Roux et al. (2005), Magoulès et al. (2004b)). At this stage, it is important to mention that the formulation of the non-overlapping Schwarz algorithm could be re-written as a preconditionned sub-structuring method.

4.1.2 Substructuring Methods

In this section, substructuring methods, i.e., methods based on nonoverlapping regions are described. This terminology comes from the structural mechanics discipline where nonoverlapping ideas were first developed. In this early work, the primary focus was on direct solvers. Associating one frontal matrix with each subdomain allows for coarse grain multiple front direct solvers, such as in (Duff et al. (1986)). Motivated by parallel distributed computing and the potential for coarse grain parallelism, considerable research activity was developed around iterative domain decomposition schemes (Farhat and Roux (1994)). A very large number of methods have been proposed and we cannot cover all of them and here only the main highlights are surveyed.
The governing idea behind sub-structuring or Schur complement methods is to split the unknowns into two subsets. This induces the following block reordered linear system :

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_I \\ b_\Gamma \end{pmatrix}, \tag{4.6}$$

where x_Γ contains all unknowns associated with subdomain interfaces and x_I contains the remaining unknowns associated with subdomain interiors. The matrix A_{II} is block diagonal where each block corresponds to a subdomain interior. Eliminating x_I from the second block row of equation (4.6) leads to the reduced system

$$Sx_\Gamma = b_\Gamma - A_{\Gamma I}A_{II}^{-1}b_I, \text{ where } S = A_{\Gamma\Gamma} - A_{\Gamma I}A_{II}^{-1}A_{I\Gamma}, \tag{4.7}$$

and S is referred to as the Schur complement matrix. This reformulation leads to a general strategy for solving (4.6). Specifically, an iterative method can be applied to (4.7). Once x_Γ is determined, x_I can be computed with one additional solve on the subdomain interiors. Further, when A is symmetric positive definite, the matrix S inherits this property and so a conjugate gradient method can be employed.
Not surprisingly, the structural analysis finite element community has been heavily involved with these techniques. Not only is their definition

fairly natural in a finite element framework but their implementation can preserve data structures and concepts already present in large engineering software packages.
Let Γ denote the entire interface with $\Gamma = \cup\Gamma_i$ where $\Gamma_i = \partial\Omega_i \setminus \partial\Omega$. As interior unknowns are no longer considered, new restriction operators must be defined as follows. Let $R_{\Gamma_i} : \Gamma \to \Gamma_i$ be the canonical pointwise restriction which maps full vectors defined on Γ into vectors defined on Γ_i. Analogous to (4.2), the Schur complement matrix (4.7) can be written as the sum of elementary matrices

$$S = \sum_{i=1}^{N} R_{\Gamma_i}^T S_i R_{\Gamma_i}, \tag{4.8}$$

where

$$S_i = A_{\Gamma_i\Gamma_i} - A_{\Gamma_i I_i} A_{I_i I_i}^{-1} A_{I_i\Gamma_i} \tag{4.9}$$

is a local Schur complement and is defined in terms of submatrices from the local matrix A_i given by (4.1). Note that this form of the Schur complement has only one layer of interface unknowns between subdomains and allows for a straight-forward parallel implementation.
While the Schur complement system is significantly better conditioned than the original matrix A , it is important to consider further preconditioning when employing a Krylov method. It is well- known, for example, that $K(A) = \mathcal{O}(h^{-2})$ when A corresponds to a standard discretization (e.g. piecewise linear finite elements) of the Laplace operator on a mesh with spacing h between the grid points. Using nonoverlapping subdomains effectively reduces the condition number of the Schur complement matrix to $K(S) = \mathcal{O}(h^{-1})$ and it is possible to further improve this condition number.

The Neumann-Dirichlet preconditioner

When a symmetric constant coefficient problem is sub-divided into two non-overlapping domains such that the subdomains are exact mirror images, it follows that the Schur complement contributions from both the left and right domains are identical. That is, $S_1 = S_2$. Consequently, the inverse of either S_1 or S_2 are ideal preconditioners as the preconditioned linear system is well-conditioned, e.g. $SR_{\Gamma_1}^T S_1^{-1} R_{\Gamma_1} = 2I$. A factorization can be applied to the local Neumann problem (4.1) on Ω_1 :

$$A_1 = \begin{pmatrix} \mathbb{I}_{I_1} & 0 \\ A_{I_1\Gamma_1} A_{I_1 I_1}^{-1} & \mathbb{I}_{\Gamma_1} \end{pmatrix} \begin{pmatrix} A_{I_1 I_1} & 0 \\ 0 & S_1 \end{pmatrix} \begin{pmatrix} \mathbb{I}_{I_1} & A_{I_1 I_1}^{-1} A_{\Gamma_1 I_1} \\ 0 & \mathbb{I}_{\Gamma_1} \end{pmatrix},$$

to obtain

$$S_1^{-1} = (0 \quad \mathbb{I}_{\Gamma_1})(A_1)^{-1} \begin{pmatrix} 0 \\ \mathbb{I}_{\Gamma_1} \end{pmatrix}.$$

In general, most problems will not have mirror image subdomains and so $S_1 \neq S_2$. However, if the underlying systems within the two subdomains are similar, the inverse of S_1 should make an excellent preconditioner. The corresponding linear system is

$$(\mathbb{I} + R_{\Gamma_1}^T S_1^{-1} R_{\Gamma_1} S_2) x_\Gamma = R_{\Gamma_1}^T S_1^{-1} R_{\Gamma_1} (b_\Gamma - A_{\Gamma I} A_{II}^{-1} b_I),$$

so that each Krylov iteration solves a Dirichlet problem on Ω_2 (to apply S_2) followed by a Neumann problem on Ω_1 to invert S_1. The Neumann-Dirichlet preconditioner is introduced in Toselli and Widlund (2005).
Generalization of the Neumann-Dirichlet preconditioner to multiple domains can be done easily when a red-black coloring of subdomains is possible such that subdomains of the same color do not share an interface. In this case, the preconditioner is just the sum of the inverses corresponding to the black subdomains :

$$M = \sum_{i \in B} R_{\Gamma_i}^T (S_i)^{-1} R_{\Gamma_i}, \tag{4.10}$$

where B corresponds to the set of all black subdomains.

The Neumann-Neumann preconditioner

Similar to the Neumann-Dirichlet method, the Neumann-Neumann preconditioner implicitly relies on the similarity of the Schur complement contribution from different subdomains. In the Neumann-Neumann approach the preconditioner is simply the weighted sum of the inverse of the S_i. In the two mirror image subdomains case, the preconditioner matrix is defined as

$$M_{NN} = \frac{1}{2}(R_{\Gamma_1}^T S_1^{-1} R_{\Gamma_1} + R_{\Gamma_2}^T S_2^{-1} R_{\Gamma_2}).$$

This motivates using the following preconditioner with multiple subdomains :

$$M_{NN} = \sum_{i=1}^{N} R_{\Gamma_i}^T D_i (S_i)^{-1} D_i R_{\Gamma_i}, \tag{4.11}$$

where D_i are the diagonal weighting matrices corresponding to a partition of unity. That is,

$$\sum_{i=1}^{N} R_{\Gamma_i}^T D_i R_{\Gamma_i} = \mathbb{I}_\Gamma.$$

The simplest choice for D_i is the diagonal matrix with entries equal to the inverse of the number of subdomains to which an unknown belongs. The Neumann-Neumann preconditioner was first discussed in Bourgat et al. (1989) and further studied in Tallec et al. (1991) where different choices for weight matrices are discussed. It should be noted that the matrices S_i can be singular for internal subdomains because they correspond to pure Neumann problems. The Moore-Penrose pseudo-inverse is often used to the local Schur complement inverse in (4.11) but other choices are possible such as inverting $A_i + \epsilon\mathbb{I}$ where ϵ is a small shift.
The Neumann-Neumann preconditioner is very attractive from a parallel implementation point of view. In particular, all interface unknowns are treated similarly and no distinction is required to differentiate between unknowns on faces, edges, or cross points as it might be the case of other approaches.

4.2 Domain Decomposition Methods for Finite Volume Methods

As mentioned in the previous section, domain decompositon methods for finite element method have been widely studied in the past decades. On the contrary, the situation is less clear for hyperbolic or mixed hyperbolic-parabolic problems using finite volume methods. These problems often arise from the modeling of compressible fluid flows.

The object of the present work is to solve the compressible fluids (both single-phase and two-phase flows) by a nonoverlapping domain decomposition method, and more precisely by a Schur complement method. A simple attempt is to adapt the principle of the domain decomposition method for elliptic problems to our problems. As in the case of elliptic problems, the principle is that we decompose the global problem into independent subproblems which are solved by each processor. More precisely, assume that we want to solve the problem :

$$\begin{cases} \dfrac{\partial \mathbf{U}}{\partial t} + \mathbf{F}^{conv}(\mathbf{U}) + \mathbf{F}^{diff}(\mathbf{U}) = 0 \quad \text{in} \quad \Omega \\ B\mathbf{U} = g \quad \text{on} \quad \partial\Omega \end{cases} \tag{4.12}$$

by a domain decomposition method with a partition of the computational domain $\Omega = \cup_{I=1}^{N} \Omega_I$ and where $\mathbf{F}^{conv}(\mathbf{U})$ and $\mathbf{F}^{diff}(\mathbf{U})$ are respectively the convection and diffusion terms of the system. Let $\mathbf{U}_I^n$ be an approximation of the solution $\mathbf{U}$ in the subdomain Ω_I at time step n. The algorithm of the domain decomposition method by Schur complement is written as

$$\begin{cases} \dfrac{\partial \mathbf{U}_I}{\partial t} + \mathbf{F}^{conv}(\mathbf{U}_I) + \mathbf{F}^{diff}(\mathbf{U}_I) = 0 \quad \text{in} \quad \Omega_I \\ B\mathbf{U}_I = g \quad \text{on} \quad \partial\Omega \cap \partial\Omega_I \\ C_I\mathbf{U}_I = C_I\mathbf{U}_J \quad \text{on} \quad \partial\Omega_I \cap \partial\Omega_J \end{cases} \tag{4.13}$$

where C_I is an interface operator which we will clarify later.

However, the implementation of these ideas in hyperbolic problems raises some technical difficulties such as :

- The scheme must be conservative ;
- In the finite volume formulation, there is no unknown defined at the interface ;
- The boundary condition of the hyperbolic systems must depend on the characteristics of the problem.

Those difficulties are solved in Dolean and Lanteri (2001) for the Euler equations by replacing the interface variables in the context of elliptic problems by the interface fluxes in the context of hyperbolic problems (4.2.3).

4.2.1 Explicit Coupling

We recall the explicit scheme (3.16)

$$\frac{\mathbf{U}_i^{n+1}-\mathbf{U}_i^n}{\Delta t}+\sum_{j\in N(i)}\frac{s_{ij}}{v_i}\{A^-(\mathbf{U}_{Roe}^n)+D(\mathbf{U}_{diff}^n)\}(\mathbf{U}_j^n-\mathbf{U}_i^n)=0 \qquad (4.14)$$

where $\mathbf{U}_i^n$ is the numerical approximation over the cell C_i at n^{th} time step. $\mathbf{U}_{Roe}^n$, $\mathbf{U}_{diff}^n$ are respectively the Roe-averaged state and the interface diffusion state of the value of $\mathbf{U}^n$ in the two neigboring cells C_i and C_j. The matrices A^- and D are the aproximations of the jacobian matrices of the convection and diffusion parts.

We would like to solve (4.14) on N processors and each processor works on one subdomain. Then, for the cell i, which belongs to the subdomain I, neighboring of the cell j, which belongs to the subdomain J, the only missing term is $\left[A^-(\mathbf{U}_{Roe}^n)+D(\mathbf{U}_{diff}^n)\right]\mathbf{U}_j^n$ which is known by the neighboring domain. So processor I needs information from processor J in order to build $\mathbf{U}_{Roe}^n$, $\mathbf{U}_{diff}^n$ and $\mathbf{U}_j^n$. We use MPI (message passing interface) to exchange the information and then all processors can work simultaneously.

4.2.2 Implicit Coupling

We recall the implicit scheme (3.17) :

$$\frac{\mathbf{U}_i^{n+1}-\mathbf{U}_i^n}{\Delta t}+\sum_{j\in N(i)}\frac{s_{ij}}{v_i}\{A^-(\mathbf{U}_{Roe}^{n+1})+D(\mathbf{U}_{diff}^{n+1})\}(\mathbf{U}_j^{n+1}-\mathbf{U}_i^{n+1})=0 \quad (4.15)$$

To solve this nonlinear system, we use the Newton method (sec. 1.4.3). Then, at each Newton iteration, we need to solve the following linear system

$$\begin{aligned}\frac{\delta\mathbf{U}^{k+1}}{\Delta t}+ & \sum_{j\in N(i)}\frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k)+D(\mathbf{U}_{diff}^k)\right]\left(\delta\mathbf{U}_j^{k+1}-\delta\mathbf{U}_i^{k+1}\right) \qquad (4.16)\\ = & -\frac{\mathbf{U}_i^k-\mathbf{U}_i^n}{\Delta t}-\sum_{j\in N(i)}\frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k)+D(\mathbf{U}_{diff}^k)\right](\mathbf{U}_j^k-\mathbf{U}_i^k)\end{aligned}$$

where $\delta\mathbf{U}^{k+1}$ denotes the variation of the k^{th} iteration that approximates the solution of time step $n+1$. As in the case of the explicit scheme, we would like to solve (4.16) on N processors and each processor work on one subdomain. We see that it lacks $\delta\mathbf{U}_j^{k+1}$ to the computational unit of the subdomain I if the cell j belongs to another subdomain, and it is not calculable by the system since $\delta\mathbf{U}_j^{k+1}$ is to be calculated. Then the processor I needs from the processor J the value $\delta\mathbf{U}_j^{k+1}$ which is not yet available. Conversely, the processor J needs $\delta\mathbf{U}_i^{k+1}$ from the processor I.

4.2.3 Dolean's Interface Variable

In the article of Dolean and Lanteri (2001), the authors do not consider the diffusion term, then the system (4.16) becomes

$$\frac{\delta \mathbf{U}^{k+1}}{\Delta t} + \sum_{j\in N(i)} \frac{s_{ij}}{v_i} A^-(\mathbf{U}^k_{Roe}) \left(\delta \mathbf{U}^{k+1}_j - \delta \mathbf{U}^{k+1}_i\right) \tag{4.17}$$
$$= -\frac{\mathbf{U}^k_i - \mathbf{U}^n_i}{\Delta t} - \sum_{j\in N(i)} \frac{s_{ij}}{v_i} A^-(\mathbf{U}^k_{Roe})(\mathbf{U}^k_j - \mathbf{U}^k_i).$$

Then, in order to obtain independent subsystems, Dolean et al. introduce a redundant flux variable $\delta\phi^{Do}_{ij}$ at the domain interface between two neighboring cells C_i and C_j which belong to different subdomains :

$$\delta\phi^{Do}_{ij} = A^+(\mathbf{U}^k_{Roe})\delta\mathbf{U}^{k+1}_i - A^-(\mathbf{U}^k_{Roe})\delta\mathbf{U}^{k+1}_j \tag{4.18}$$

where A^+ is defined as in (1.18). Then by defining the orthogonal projectors $P^{\pm}_{Do}(\mathbf{U}^k_{Roe})$ on the eigenvector subspaces such that :

$$P^-_{Do}(\mathbf{U}^k_{Roe})\delta\phi^{Do}_{ij} = A^-(\mathbf{U}^k_{Roe})\delta\mathbf{U}^{k+1}_j,\ P^+_{Do}(\mathbf{U}^k_{Roe})\delta\phi^{Do}_{ij} = -A^+(\mathbf{U}^k_{Roe})\delta U^{k+1}_i.$$

The system (4.17) can be written as

$$\frac{\delta \mathbf{U}^{k+1}}{\Delta t} + \sum_{j\in I, j\in N(i)} \frac{s_{ij}}{v_i} A^-(\mathbf{U}^k_{Roe}) \left(\delta \mathbf{U}^{k+1}_j - \delta \mathbf{U}^{k+1}_i\right) - \sum_{j\in I, j\in N(i)} \frac{s_{ij}}{v_i} A^-(\mathbf{U}^k_{Roe})\delta\mathbf{U}^{k+1}_i \tag{4.19}$$
$$= -\frac{\mathbf{U}^k_i - \mathbf{U}^n_i}{\Delta t} - \sum_{j\in N(i)} \frac{s_{ij}}{v_i} A^-(\mathbf{U}^k_{Roe})(\mathbf{U}^k_j - \mathbf{U}^k_i) - \sum_{j\notin I, j\in N(i)} \frac{s_{ij}}{v_i} P^-_{Do}(\mathbf{U}^k_{Roe})\phi^{Do}_{ij}.$$

By defining $\mathbf{U}_I = (\mathbf{U}_1, \ldots, \mathbf{U}_m)^t$ the unknown vector of the subdomain I and

$$\delta\phi_I = (\delta\phi_{ij})_{i\in I, j\in J, j\in N(i)} \tag{4.20}$$

and by denoting $\mathcal{A}_I$ the local Neumann matrix of the subdomain I and P_I all the projectors P^-_{Do} in the subdomain I, they can write the linear system as

$$\mathcal{A}(\mathbf{U}^k)\delta\mathbf{U}^{k+1}_I = b_I(\mathbf{U}^n, \mathbf{U}^k) - P_I\delta\phi^{Do}_I \tag{4.21}$$

By taking into account Eqs. (4.18), (4.20) and (4.21), and denoting

$$\delta\Phi^{Do} = (\delta\phi^{Do}_I),\ I = 1 \ldots N,$$

they obtain an extended system that distinguishes the internal unknowns from the interface ones :

$$\left(\begin{array}{cccc|c} \mathcal{A}_1 & 0 & \ldots & \ldots & P_1 \\ 0 & \mathcal{A}_2 & 0 & \ldots & P_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \mathcal{A}_N & P_N \\ \hline M_1 & \ldots & \ldots & M_N & \mathbb{I} \end{array}\right) \left(\begin{array}{c} \delta\mathbf{U}_1 \\ \delta\mathbf{U}_2 \\ \ldots \\ \delta\mathbf{U}_N \\ \delta\Phi^{Do} \end{array}\right) = \left(\begin{array}{c} b_1 \\ b_2 \\ \ldots \\ b_N \\ b^{Do}_\Phi \end{array}\right) \tag{4.22}$$

where $\mathcal{A}_I$ is the matrix that couples the unknowns associated with internal cells of Ω_I whereas M_I links $\delta\mathbf{U}_I$ to $\delta\Phi^{Do}$ through (4.18).

The internal unknowns in (4.22) can be eliminated in favor of the interface ones to yield the following interface system :

$$S\delta\Phi^{Do} = b_\Phi^{Do} \tag{4.23}$$

with

$$\begin{aligned}(S\delta\Phi^{Do}) &= \delta\Phi^{Do} + \sum_{I=1}^{N} M_I {\mathcal{A}_I}^{-1} P_I \delta\phi_I^{Do} \\ (b_\Phi^{Do}) &= \sum_{I=1}^{N} M_I {A_I}^{-1} b_I\end{aligned}$$

Unfortunately, this strategy can only be applied to the Euler equations using the upwind scheme.

4.2.4 A New Interface Variable

In order to include diffusion terms in the model and to use various schemes and various systems, we introduce a new interface flux variable $\delta\phi_{ij}$ at the domain interface between two neighboring cells C_i and C_j which belong to different subdomains :

$$\delta\phi_{ij} = \delta\mathbf{U}_j - \delta\mathbf{U}_i \tag{4.24}$$

In the case where the cell i of the subdomain I is at the boundary and has to communicate with the neighboring subdomains, we can rewrite the system (4.16) as :

$$\begin{aligned}\frac{\delta\mathbf{U}_i^{k+1}}{\Delta t} + \sum_{j\in I, j\in N(i)} \frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k) + D(\mathbf{U}_{diff}^k)\right]\left(\delta\mathbf{U}_j^{k+1} - \delta\mathbf{U}_i^{k+1}\right) \\ = -\frac{\mathbf{U}_i^k - \mathbf{U}_i^n}{\Delta t} - \sum_{j\in N(i)} \frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k) + D(\mathbf{U}_{diff}^k)\right](\mathbf{U}_j^k - \mathbf{U}_i^k) \\ - \sum_{j\notin I, j\in N(i)} \frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k) + D(\mathbf{U}_{diff}^k)\right]\delta\phi_{ij}^{k+1}\end{aligned}$$

As similar to the previous section, we define $\mathbf{U}_I = (\mathbf{U}_1, \ldots, \mathbf{U}_m)^t$ the unknown vector of the subdomain I and

$$\delta\phi_I = (\delta\phi_{ij})_{i\in I, j\in J, j\in N(i)}. \tag{4.25}$$

We also denote $\mathcal{A}_I$ the local Neumann matrix of the subdomain I and $P_I = \sum_{j\notin I, j\in N(i)} \frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k) + D(\mathbf{U}_{diff}^k)\right]$, we can write the linear system as :

$$\mathcal{A}_I(\mathbf{U}^k)\delta\mathbf{U}_I^{k+1} = b_I(\mathbf{U}^n, \mathbf{U}^k) - P_I\delta\phi_I \tag{4.26}$$

By taking into account Eqs. (4.24), (4.25) and (4.26),and denoting

$$\delta\Phi = (\delta\phi_I),\ I = 1 \ldots N$$

we can build an extended system that distinguishes the internal unknowns from the interface ones :

$$\left(\begin{array}{cccc|c} \mathcal{A}_1 & 0 & \dots & \dots & P_1 \\ 0 & \mathcal{A}_2 & 0 & \dots & P_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \mathcal{A}_N & P_N \\ \hline M_1 & \dots & \dots & M_N & \mathbb{I} \end{array}\right) \left(\begin{array}{c} \delta\mathbf{U}_1 \\ \delta\mathbf{U}_2 \\ \dots \\ \delta\mathbf{U}_N \\ \delta\Phi \end{array}\right) = \left(\begin{array}{c} b_1 \\ b_2 \\ \dots \\ b_N \\ b_\Phi \end{array}\right) \tag{4.27}$$

where $\mathcal{A}_I$ is the matrix that couples the unknowns associated with internal cells of Ω_I whereas M_I links $\delta\mathbf{U}_I$ to $\delta\Phi$ through (4.24). Then, in this case, M_I comprises only 0 or ± 1.

The internal unknowns in (4.27) can be eliminated in favor of the interface ones to yield the following interface system :

$$S\delta\Phi = b_\Phi \tag{4.28}$$

with

$$\begin{aligned} (S\delta\Phi) &= \delta\Phi + \sum_{I=1}^{N} M_I {\mathcal{A}_I}^{-1} P_I \delta\phi_I \\ (b_\Phi) &= \sum_{I=1}^{N} M_I {A_I}^{-1} b_I \end{aligned}$$

The computation of the matrix S is so costly as we have to inverse the local matrix $\mathcal{A}_I$. Fortunately, we do not have to compute explicitly the coefficients of S. All we need is to design the operator $\delta\Phi \rightarrow S\delta\Phi$. Then the equation (4.28) can be solved by, e.g., GMRES, BICGStab, or the Richardson methods.

Once we have solved the interface system, we know $\delta\Phi$ and then we can solve the internal unknowns on each processor using the equation 4.26.
We now present the procedure of the resolution of our interface system.

Implementation

We first initialize $\delta\Phi = \mathbf{0}$. Then, we build the right-hand-side $b_\Phi = \sum_{I=1}^{N} b_{I,\Phi}$ with $b_{I,\Phi} = M_I x_I$ where x_I is obtained through the local solution : $\mathcal{A}_I x_I = b_I$.

Then, given $\delta\Phi$ we need to build the matrix-vector product $S\delta\Phi$. This procedure can be performed by solving the local system 4.26. We define $R_I(\delta\Phi)$ the solution of 4.26. Then, we have :

$$\begin{aligned} \mathcal{A}_I^{-1} b_I &= R_I(\mathbf{0}) \\ \mathcal{A}_I^{-1} \left(P_I \delta\phi_I\right) &= R_I(\mathbf{0}) - R_I(\delta\Phi) \end{aligned} \tag{4.29}$$

From (4.29), we deduce :

$$(S\delta\Phi) = \delta\Phi + \sum_{I=1}^{N} M_I \left(R_I(\mathbf{0}) - R_I(\delta\Phi)\right) \tag{4.30}$$

Equation (4.30) shows that we can obtain the matrix-vector product $(S\delta\Phi)$ once we have known the local solution $R_I(\delta\Phi)$. We do not need to compute S explicitly and we have done that in our codes.

Interface System

A New Boundary Condition As we have introduced a new interface flux variable, we need to add that variable to the code. When the cell is at the interface between two neighboring subdomains, according to (4.26), we need to add

$$-P_I \delta\phi_I$$

to the right hand side of the system.
Since $P_I = \sum_{j \notin I, j \in N(i)} \frac{s_{ij}}{v_i} \left[A^-(\mathbf{U}^k_{Roe}) + D(\mathbf{U}^k_{diff}) \right]$ which contains all the values at the previous step and $\delta\phi_I$ is a given values, the computation of $-P_I \delta\phi_I$ is possible.

Code Management and MatShell We solve the system $S\delta\Phi = b_\Phi$ by an iterative method such as GMRES, BICGStab etc. Fortunately, it is possible to do this using Petsc (Balay et al. (2012)) without explicitly coefficent computing S. This requires to use a special type of matrix, the MatShell, which are only defined by the operators, so we need to make sense of the product $S\delta\phi$. It means that we need to design the operator $\delta\Phi \mapsto S\delta\Phi$.

For a given processor I, we have the following algorithm to compute $S\delta\Phi$

- Receive $\delta\phi_I = (\delta\phi_{ij})_{i \in I, j \in J, j \in N(i)}$ from all neighboring processors using MPI ;
- Build and solve the local system (4.26) ;
- Take the trace on the interface which is neighboring with other subdomains (processors) ;
- Send the result for parallel addition as in the equation 4.24.

We can see that the compution of $S\delta\Phi$ can be performed in parallel, and it is an advantage of our method. Unfortunately, we have not found any way to precondition the operator. And we need to further work on this issue.

5 Domain Decomposition Method for Compressible flows

Contents

In this chapter, we will present our numerical results for the compresible Navier-Stokes equations and the isentropic two-fluid model in various 2D and 3D configurations and various schemes. The results show that our method is robust and efficient. We also study the scaling strategy (3.1.3) to improve the condition number of the interface system. Comparisons of performances with classical distributed computing with up to 512 processors are also reported.

5.1 Validation

In order to validate our methods, we compare the solutions obtained using single or multiple domains. As usual, we use a cartesian mesh as presented in Fig 5.1. We then decompose the computational domain into 2 and 4 subdomains as showed in Figs. 5.2 and 5.3

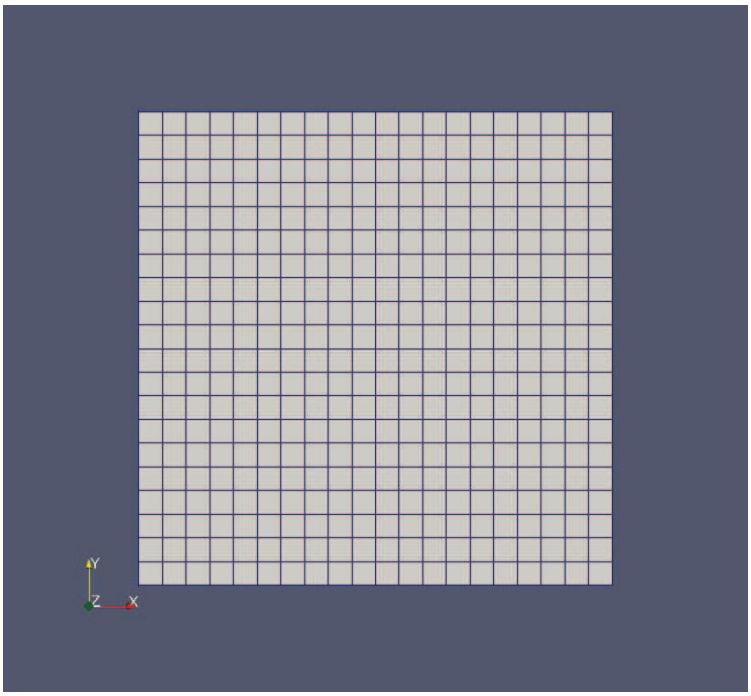

Figure 5.1 – *Mesh*

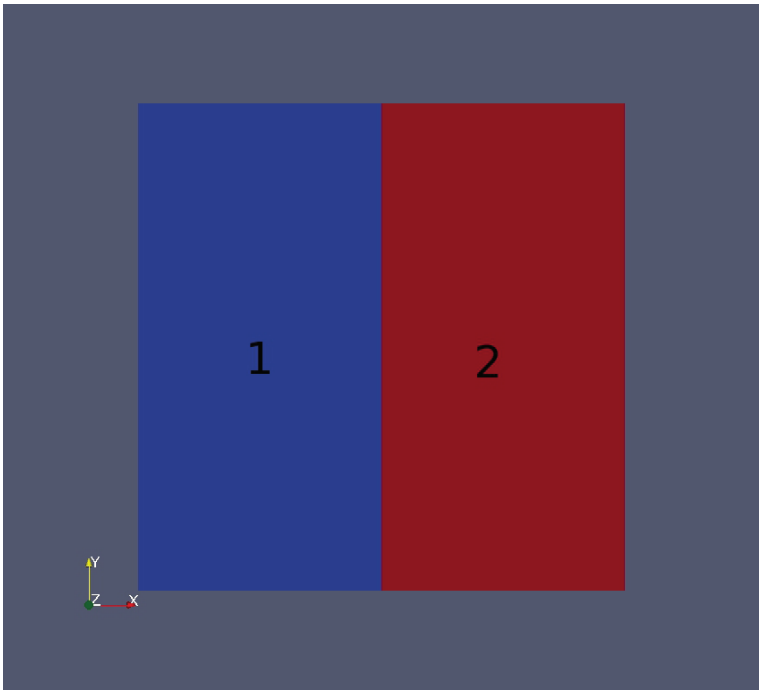

Figure 5.2 – *Processor position*

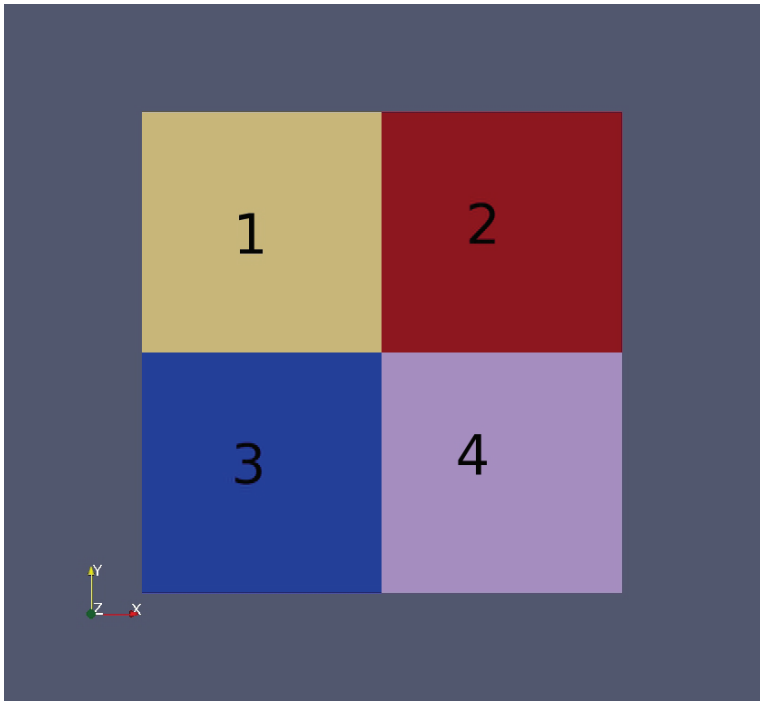

FIGURE 5.3 – *Processor position*

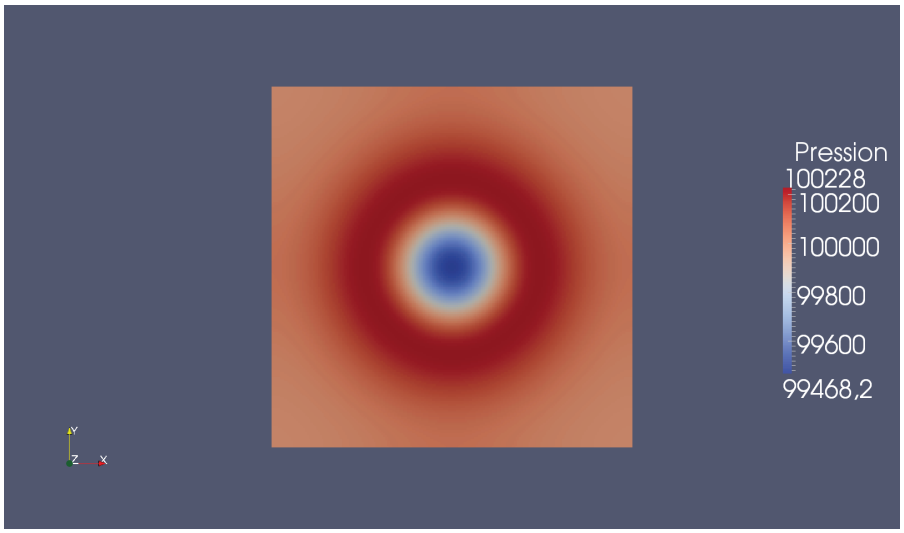

FIGURE 5.4 – *Profile of the pressure at time step* 5 *on one processor*

Figs.5.4, 5.5 and 5.6 present the profile of the pressure after 5 time steps using the upwind scheme (3.18) with CFL = 10 for the Euler equations (Equation (3.1) without viscosity and heat conductivity terms) on 1, 2 and 4 processors. Our initial state is a pressurized ball at the center of a closed box and for $t > 0$ there are waves which propagate and reflect all over the box. The gas expands in the box and we can see the shock waves and the rarefaction waves. The solution is solved on a cartesian mesh of 200×200 cells.

In Figs.5.7, 5.8 and 5.9 we show the results obtained with the same test but at time step 10. We can see that the solutions with the domain

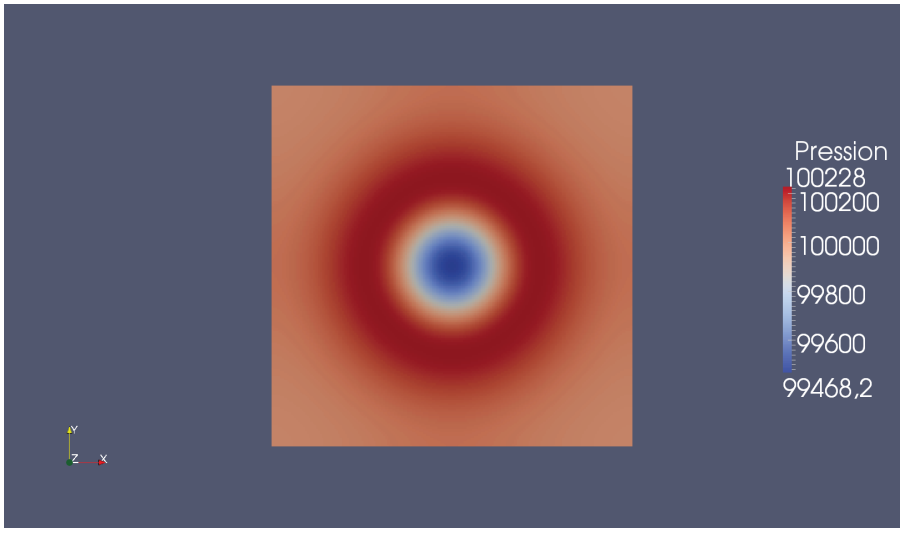

FIGURE 5.5 – *Profile of the pressure at time step* 5 *on two processors*

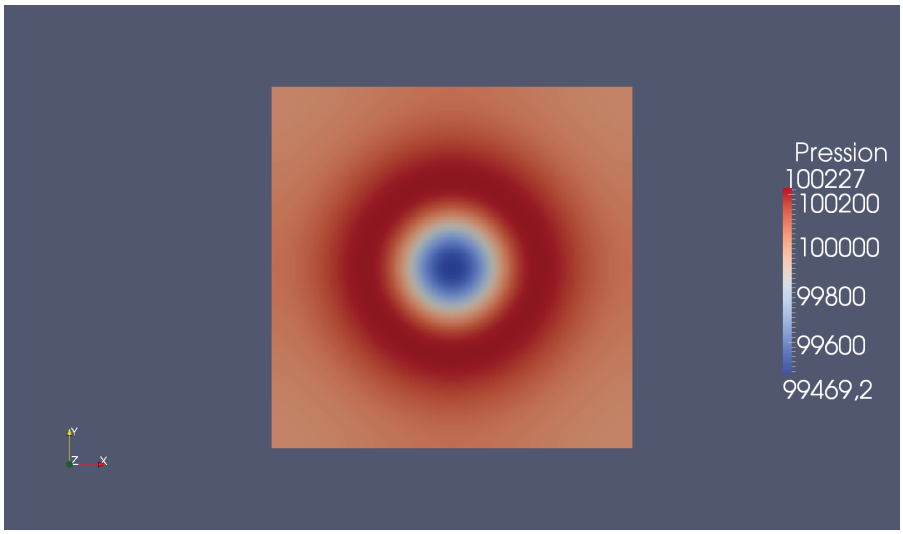

FIGURE 5.6 – *Profile of the pressure at time step* 5 *on four processors*

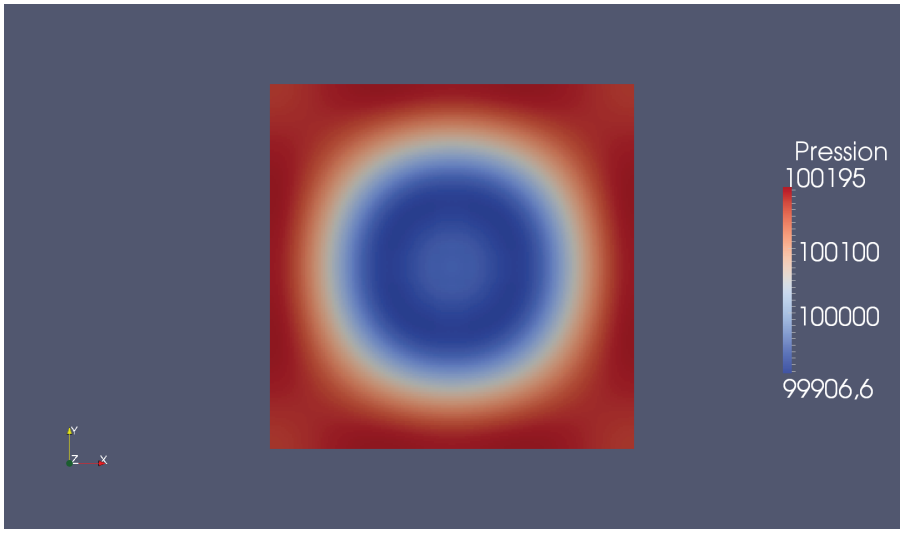

FIGURE 5.7 – *Profile of the pressure at time step* 10 *on one processor*

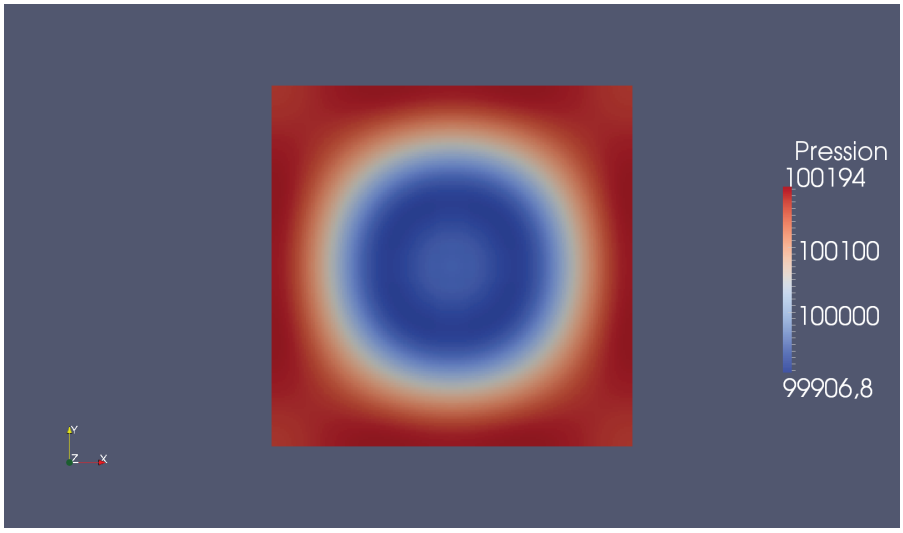

FIGURE 5.8 – *Profile of the pressure at time step* 10 *on two processors*

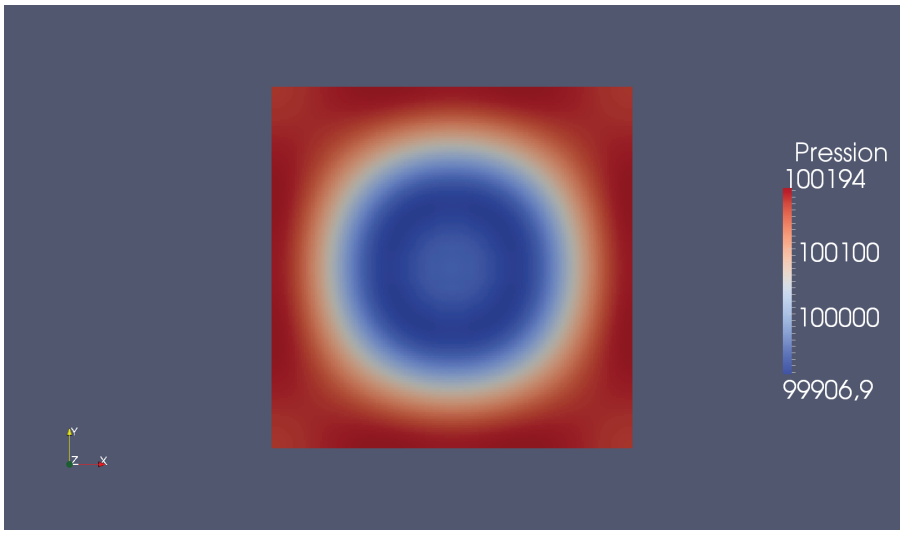

FIGURE 5.9 – *Profile of the pressure at time step* 10 *on four processors*

decomposition method are the same as that of the sequential one.

Figs. 5.10, 5.11 and 5.12 present the streamlines of the steady state obtained using the centered scheme (3.18) to solve a lid driven cavity flow at Reynolds number 400 on a cartesian 50×50 mesh. The lid speed is $1\,m/s$, the maximum Mach number of the flow is 0.008.

We then show in Figs. 5.13, 5.14 and 5.15 the results obtained with the same test but using upwind scheme. According to these results, we obtain the same solutions by using single or multiple domains.

5.2 Study of Parallel Scalability

5.2.1 Definition of Scalability

Several types of scalability are formally defined in the parallel computing literature, and numerous other operational definitions of scalability

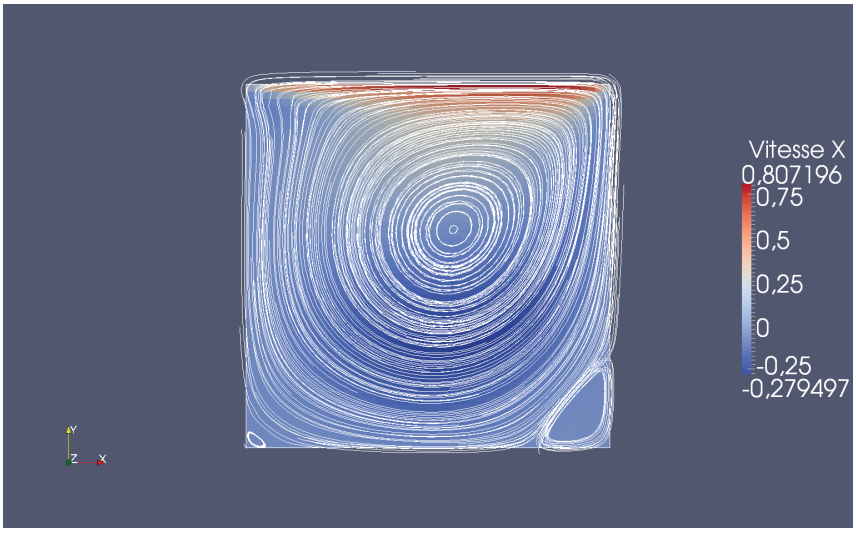

FIGURE 5.10 – *Streamlines of V_x on one processor*

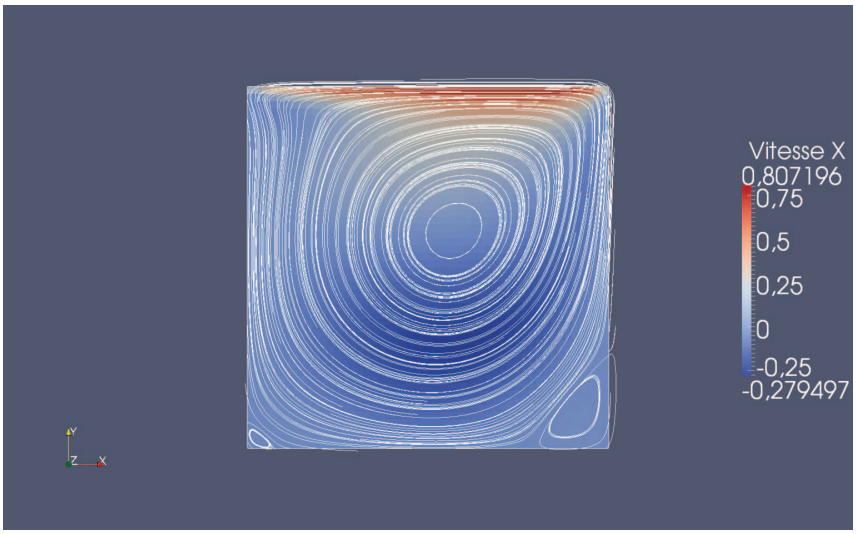

FIGURE 5.11 – *Streamlines of V_x on two processor*

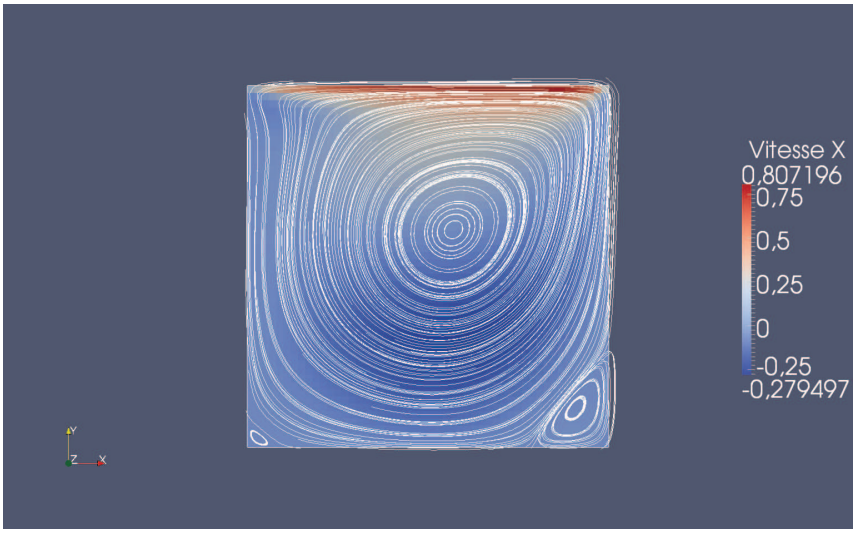

FIGURE 5.12 – *Streamlines of V_x on four processor*

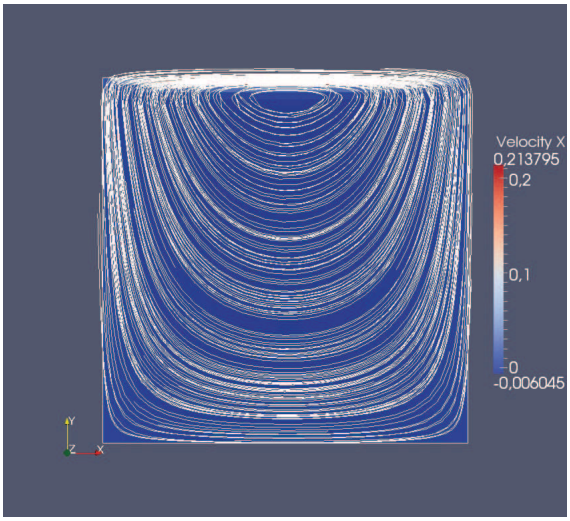

FIGURE 5.13 – *Streamlines of V_x on one processor*

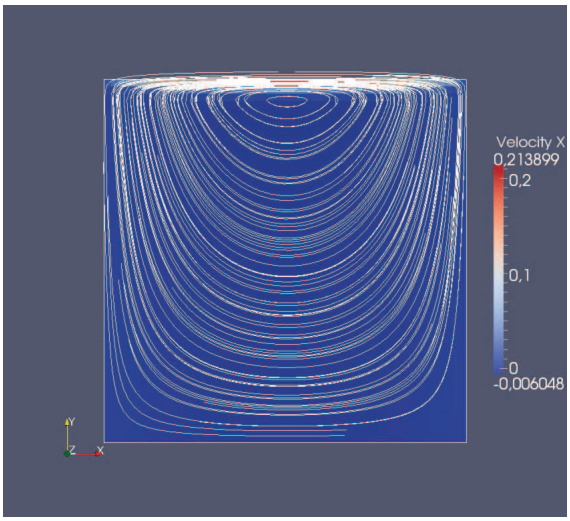

FIGURE 5.14 – *Streamlines of V_x on two processor*

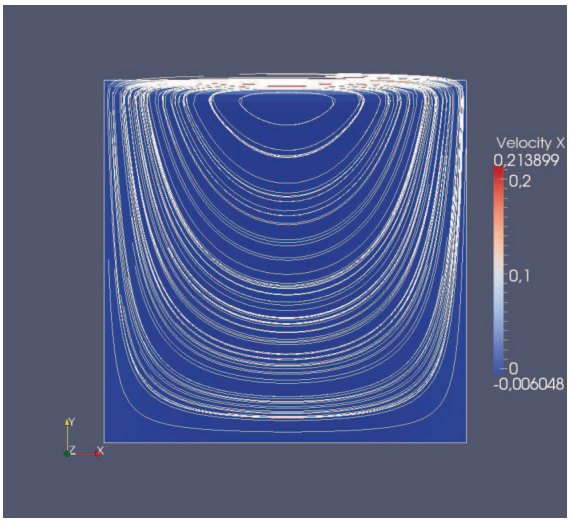

FIGURE 5.15 – *Streamlines of V_x on four processor*

exist in different scientific subcommunities, making semantic digression a necessity before any consideration of the scalability of domain decomposi-

tion methods. Given a performance metric that is a function of parameters $\{\pi_1, \pi_2, \cdots\}$, we study the variation of the metric as the π_i are varied, independently or on some constrained manifold. If the metric exhibits acceptable behavior over some region of parameter space, performance is said to "scale" over that region. Scalability of interesting numerical algorithms is rarely uniform over the entire parameter space, so claims of scalability should ordinarily be accompanied by a statement of applicable domain (Keyes (1998)).

We consider as parameters N, the numbers of cells, and P, the number of processors employed, and performance metric η, the parallel efficiency. The variation of $\eta(N, P)$ as P varies with fixed N is the "fixed-problem-size" scaling. The variation of $\eta(N, P)$ as N and P vary with N/P fixed is the "fixed- storage-per-processor" scaling. If computational work is not linear in N, then a fixed-work-per-processor scaling may be useful to consider. The expression of scalability may be inverted, e.g., the isoefficiency function, $N(P)$, of an algorithm may be sought such that $\eta(N(P), P)$ = constant, as P varies.

Fixed-storage-per-processor is arguably the most interesting limit from an architectural point of view, for three reasons. First, the proportion of memory to processors is typically fixed (at purchase). Second, work and communication are designed to scale as different powers of N/P in domain decomposition methods. If N/P varies, the ratio of communication to work varies, becoming more parasitic as N/P becomes smaller. Third, the performance of a single processor-memory unit may vary considerably with local problem size, due to cache effects. There are often thresholds of workingset size, across which performance jumps. Careful attention to data locality smooths out these thresholds, providing a range of problem size over which performance is nearly level, but if such thresholds are not controlled for, they may obscure parallel performance evaluation. Keeping N/P constant avoids the possibility of threshold effects. Despite the aesthetic superiority of fixed-storage-per-processor scaling, fixed-size scaling is often performed in practice because, for instance, grid generation is more responsive to discretization demands than parallelization opportunities. Since fixed-size scalability is more difficult to achieve (because of the second point above), we measure it, rather than fixed-storage-per-processor scalability.

Absolute efficiency on P processors is defined as the speedup divided by P

$$\eta(N, P) = \frac{1}{P} . \frac{T(N, 1)}{T(N, P)}, \tag{5.1}$$

where $T(N, P)$ is the execution time on P processors. Over smaller ranges of variation of processor number, it is useful to define the relative efficiency, in going from Q to P processors $(P > Q)$

$$\eta(N, P/Q) = \frac{Q}{P} . \frac{T(N, Q)}{T(N, P)}. \tag{5.2}$$

5.2.2 Numerical Results and Scalability

We now study the robustness and the scalability of our numerical method using the lid driven cavity test. In Figs. 5.16 and 5.17, we compare the parallel efficiency of different preconditioners on $2D$ and $3D$ computations and with $Q = 2$ and $P = 4$. We see that without the preconditioner

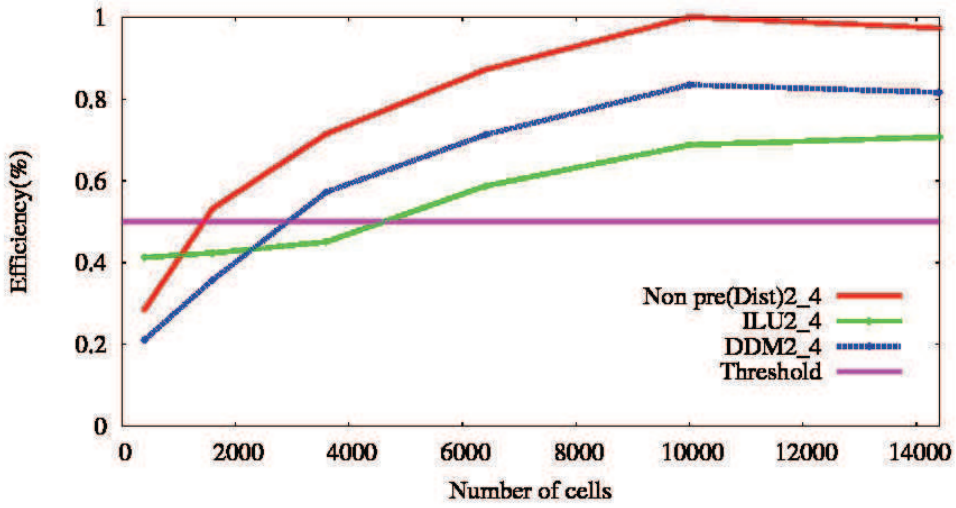

FIGURE 5.16 – *Parallel efficiency for 2D Lid driven cavity*

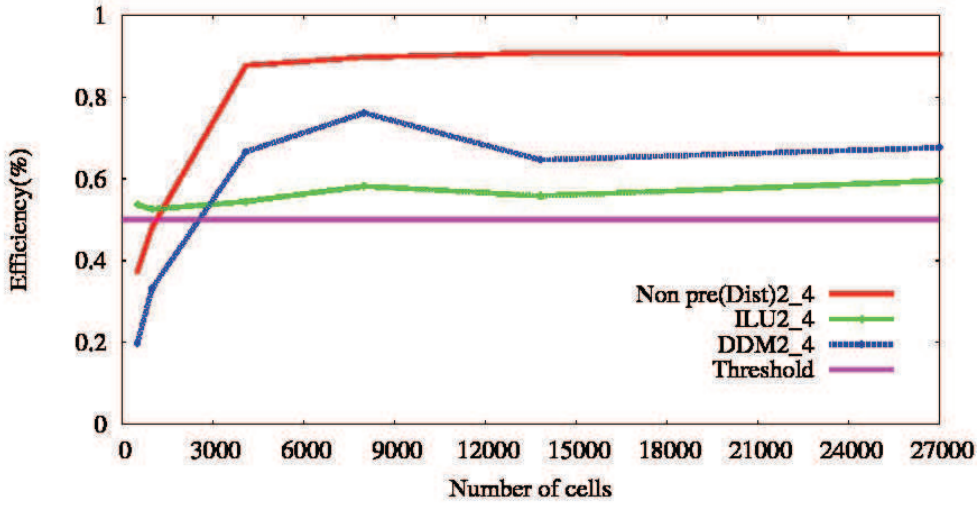

FIGURE 5.17 – *Parallel efficiency for 3D Lid driven cavity*

the solver is scalable. However, when we use the ILU preconditioner, the scalability is not optimal especially for $3D$ problems. In the two cases, our method proves better than ILU when we increase the number of cells in each subdomain. It can be also seen that we need at least 1000 cells in each subdomain to make the parallel computing rentable in comparison to the sequential one.

5.2.3 Comparison of Our Method with Dolean Method

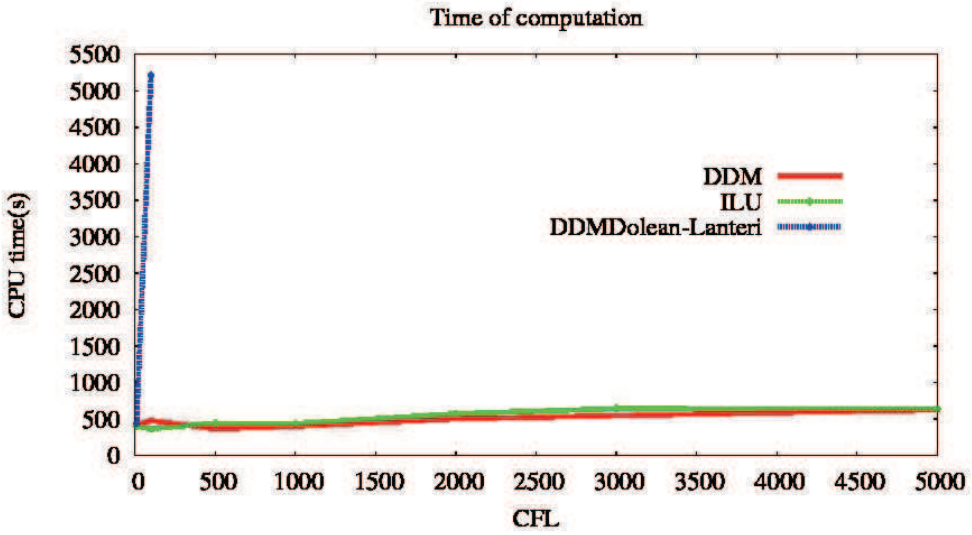

FIGURE 5.18 – *Comparisons of parallelism in 3D Detonation, global mesh* = $50 \times 50 \times 50$

In Fig. 5.18, we compare the robustness of different methods using the detonation problem. We use this problem because there is no viscosity and we can perform the computation with Dolean method. This problem is solved on a catersian $50 \times 50 \times 50$ cell mesh on two processors. The computation time of Dolean and Lanteri method increases rapidly because it requires many Newton iterations for convergence at each time step.

5.2.4 Strong Scalability

In order to assess our method, we compare the computation time of the ILU preconditioner, our method and our method with Scaling strategy (3.1.3).

Fig. 5.19 presents the computational time required to perform a time step of a fixed global problem of size one million. We compare the computational time required using the classical distributed method (red curve), the domain decomposition method (blue curve) and the domain decomposition method with scaling (green curve). We vary the number of processors up to 128. One can see that the domain decomposition method is comparable with classical distributed method and using scaling (3.1.3) is better.

In Fig 5.19 we see only two curves. This is because, in this case the classical distributed method does not converge as we use the centered scheme. Domain decomposition is the only method that converges. Similarly, we can see in Figs 5.21 and 5.22 the results obtained in the case of the two-phase flow.

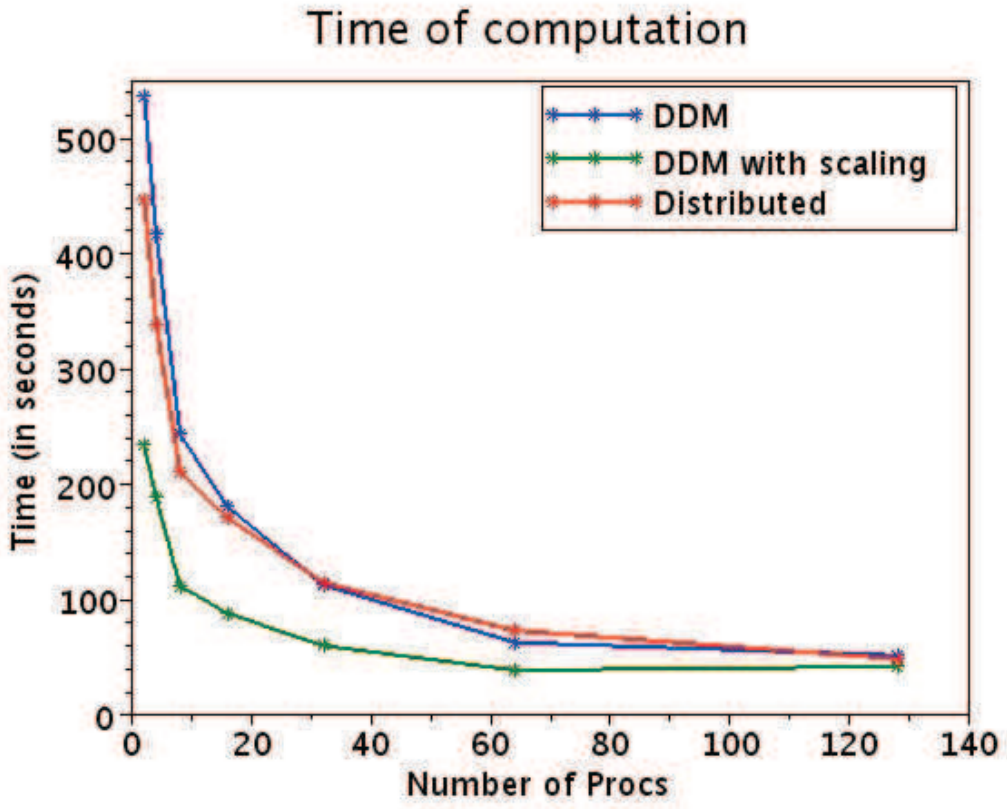

FIGURE 5.19 – *Time of computation for one time step, global mesh* = $96 \times 96 \times 96$, *CFL* 20

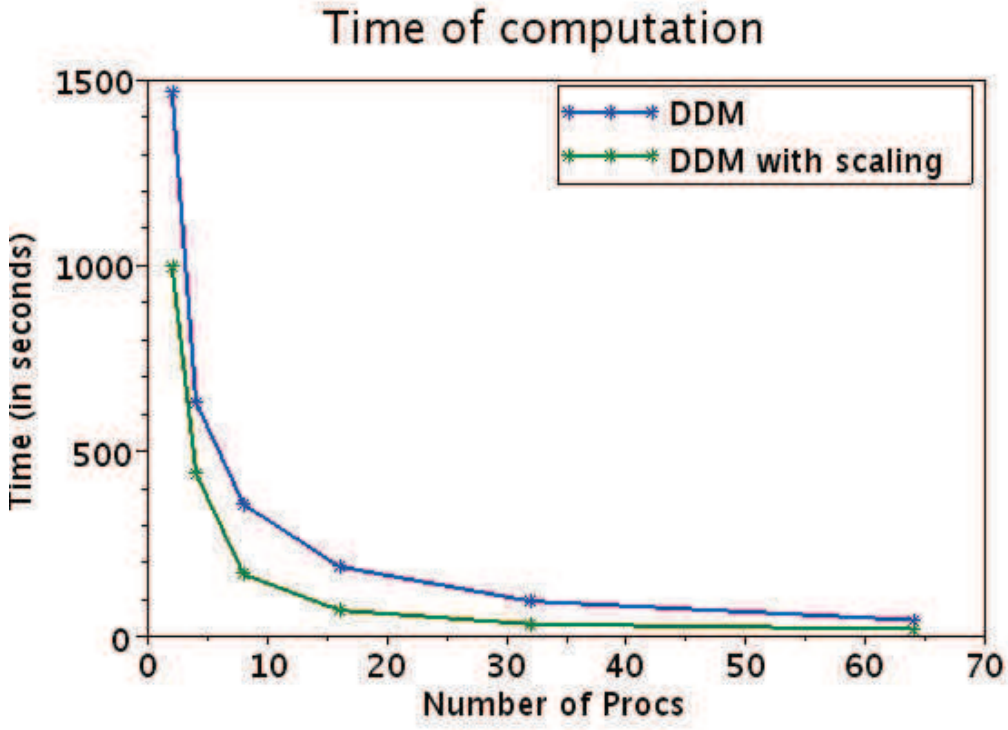

FIGURE 5.20 – *Time of computation for one time step, global mesh = 96x96x96, CFL 20*

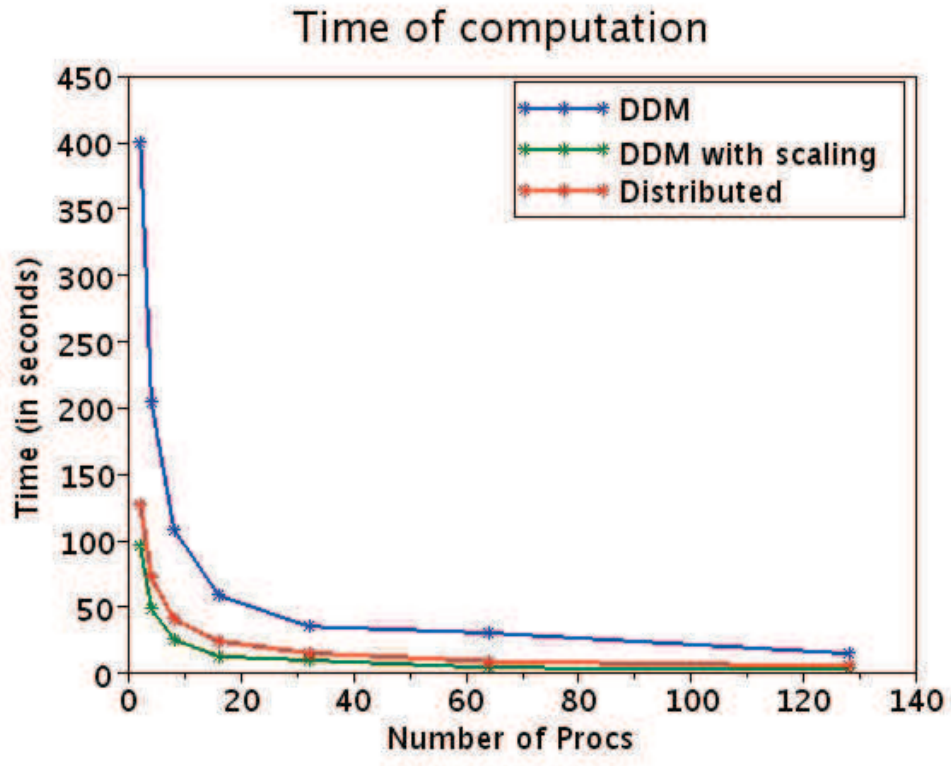

FIGURE 5.21 – *Time of computation for one time step, global mesh* = $96 \times 96 \times 96$, *CFL* 10

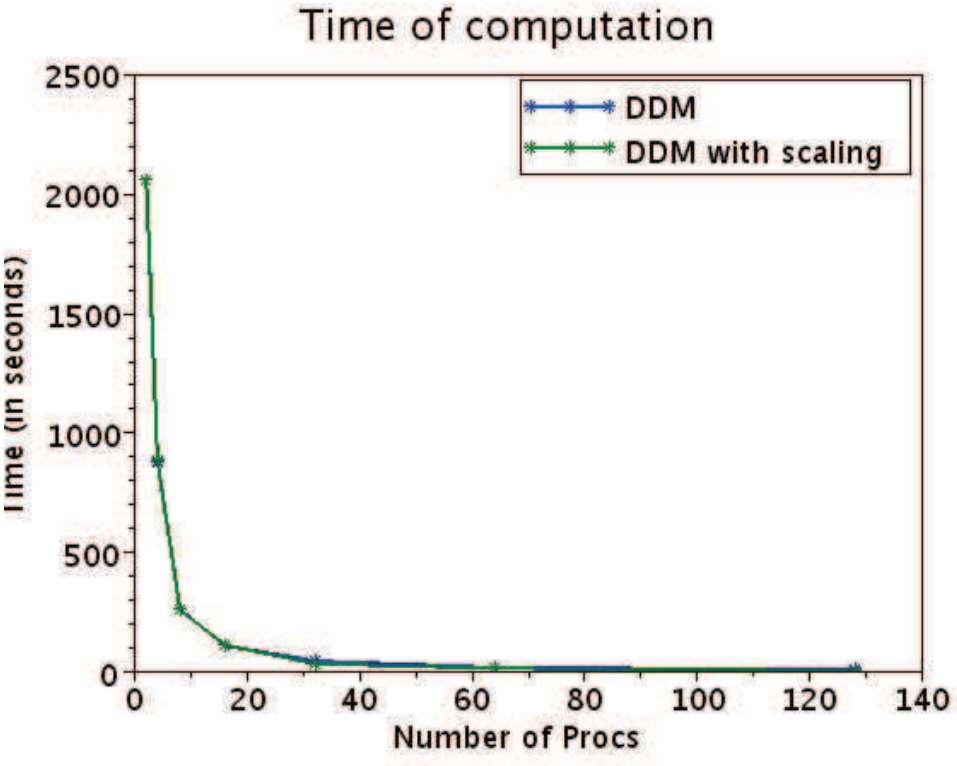

FIGURE 5.22 – *Time of computation for one time step, global mesh* = $96 \times 96 \times 96$, *CFL* 20

5.3 Study of the Spectrum of the Interface System

In the domain decomposition method it is very important to reduce the number of the interface system. Unfortunately, as we use the MatShell matrix to build the matrix S. We are not able to precondition the matrix with ILU before solving it. In order to better understand the structure of that important matrix, in this section, we propose to study its spectrum.

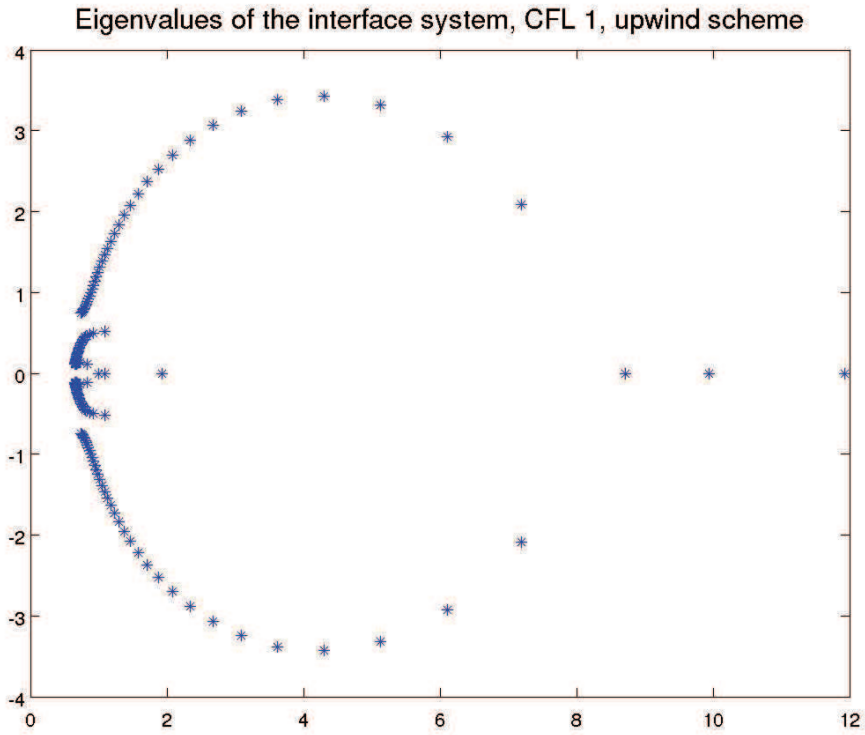

Figure 5.23 – *CFL* 1, 40×40 *cells*, 2 *processors*

In Figs. 5.23, 5.24 and 5.25 we show the spectrum of the interface system on two processors using upwind scheme with CFL = 1, 10 and 50 respectively. The global mesh is 40×40 cells. We can see that when we increase the CFL, the eigenvalues of the interface system are clustered closer to the origin. It make the interface system more difficult to solve.

Figs 5.26, 5.27 and 5.28 present the spectrum of the interface system on four processors using the upwind scheme with CFL = 1, 10 and 50 respectively. It can be seen that the spectrum of those cases is very different from the previous ones. And we see some isolated eigenvalues. If we could eliminate those eigenvalues, may be we would obtain a better solver for the interface system.

In Figs. 5.29, 5.30 and 5.31 we present the spectrum of the interface system on two processors using the centered scheme with CFL = 1, 10 and 50 respectively. The global mesh is 40×40 cells.

Figs 5.32, 5.33 and 5.34 present the spectrum of the interface system

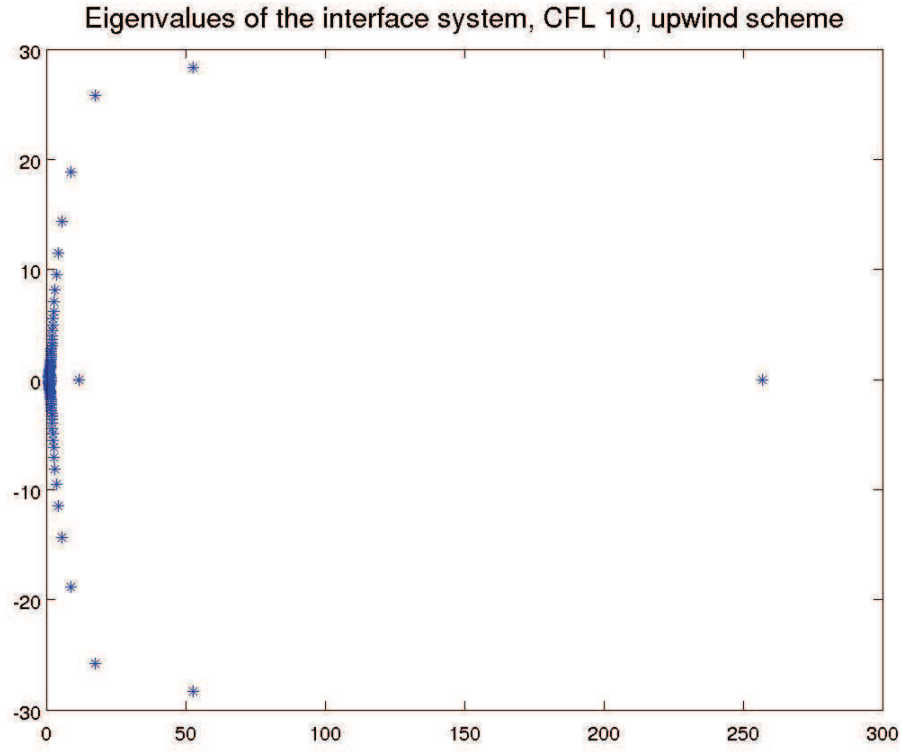

FIGURE 5.24 – *CFL* 10, 40×40 *cells*, 2 *processors*

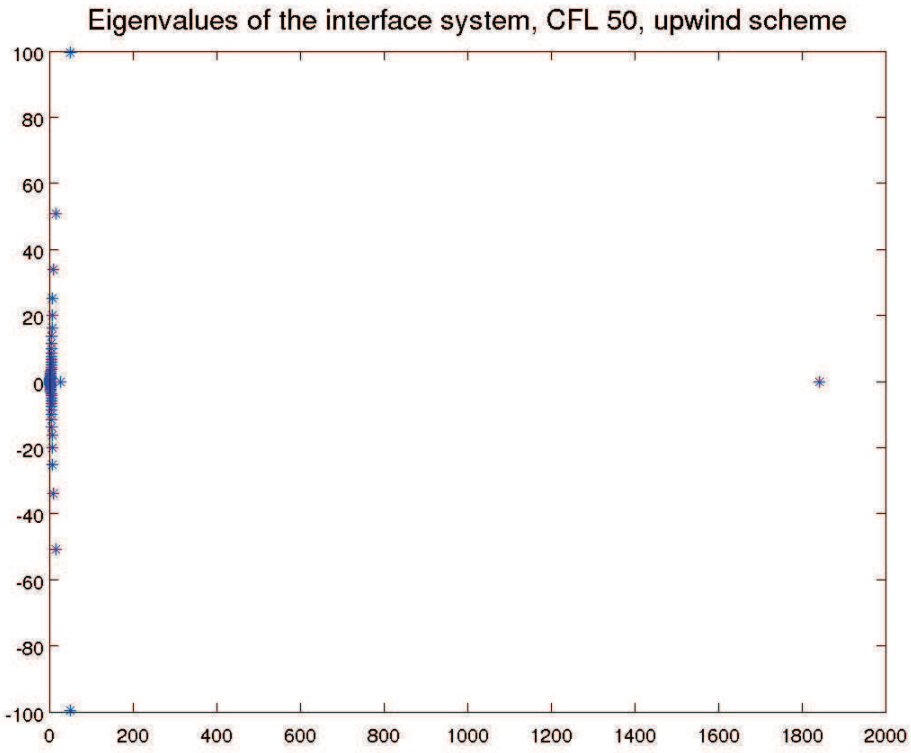

FIGURE 5.25 – *CFL* 50, 40×40 *cells*, 2 *processors*

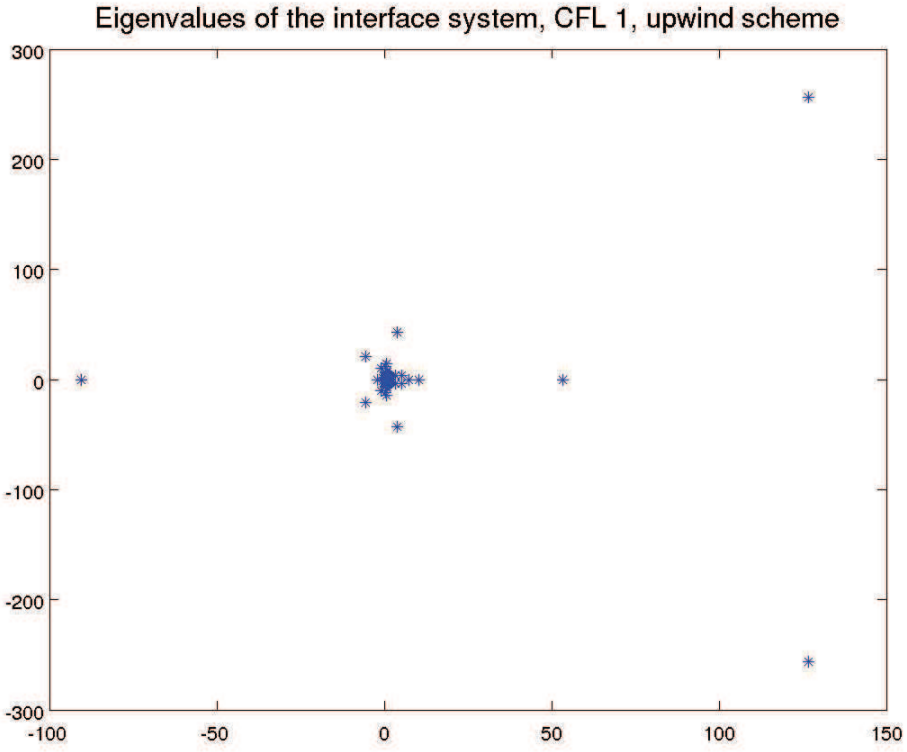

FIGURE 5.26 – *CFL* 1, 40×40 *cells*, 4 *processors*

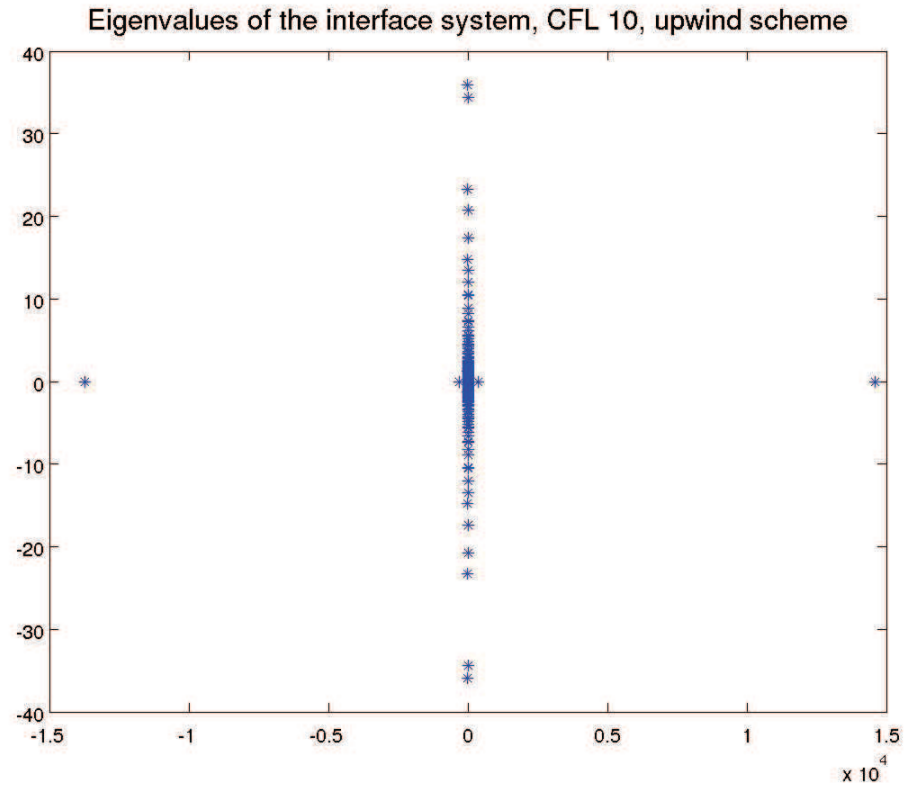

FIGURE 5.27 – *CFL* 10, 40×40 *cells*, 4 *processors*

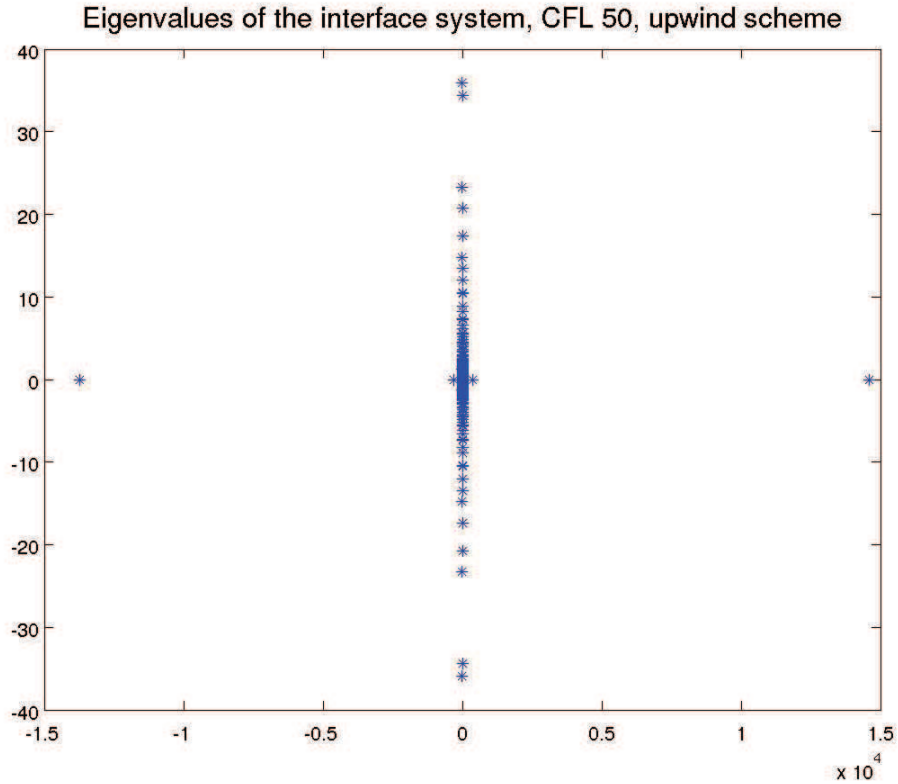

FIGURE 5.28 – *CFL* 50, 40×40 *cells*, 4 *processors*

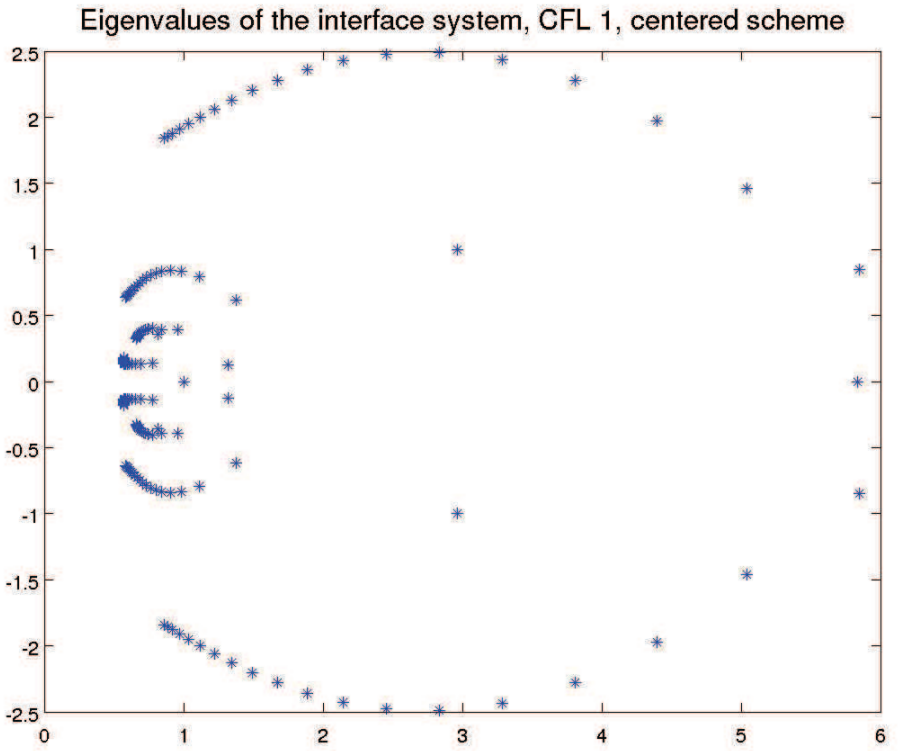

FIGURE 5.29 – *CFL* 1, 40×40 *cells*, 2 *processors*

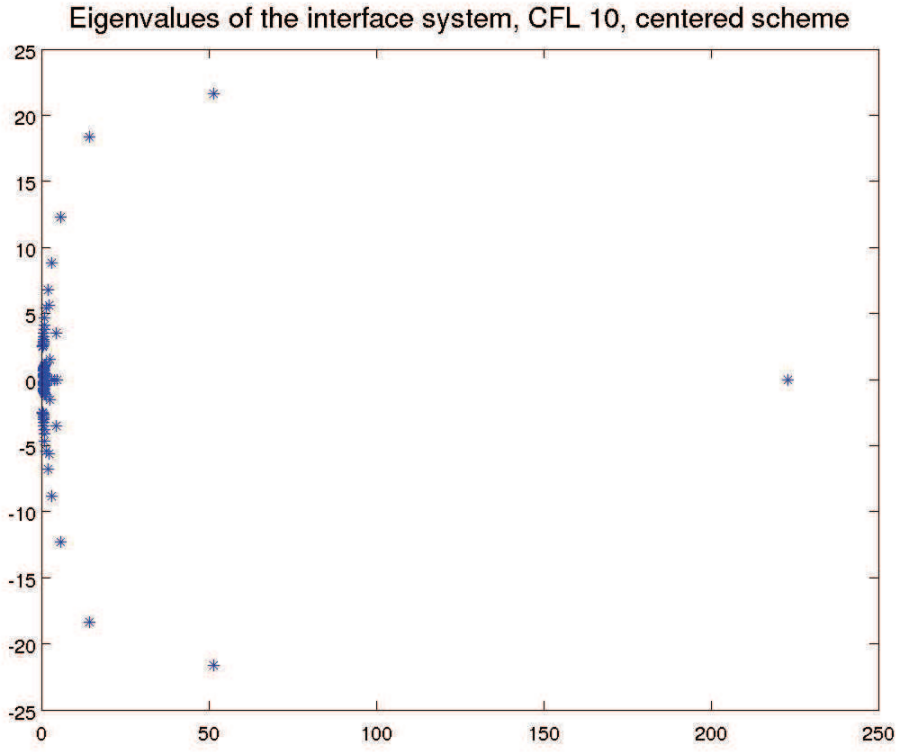

FIGURE 5.30 – *CFL* 10, 40×40 *cells*, 2 *processors*

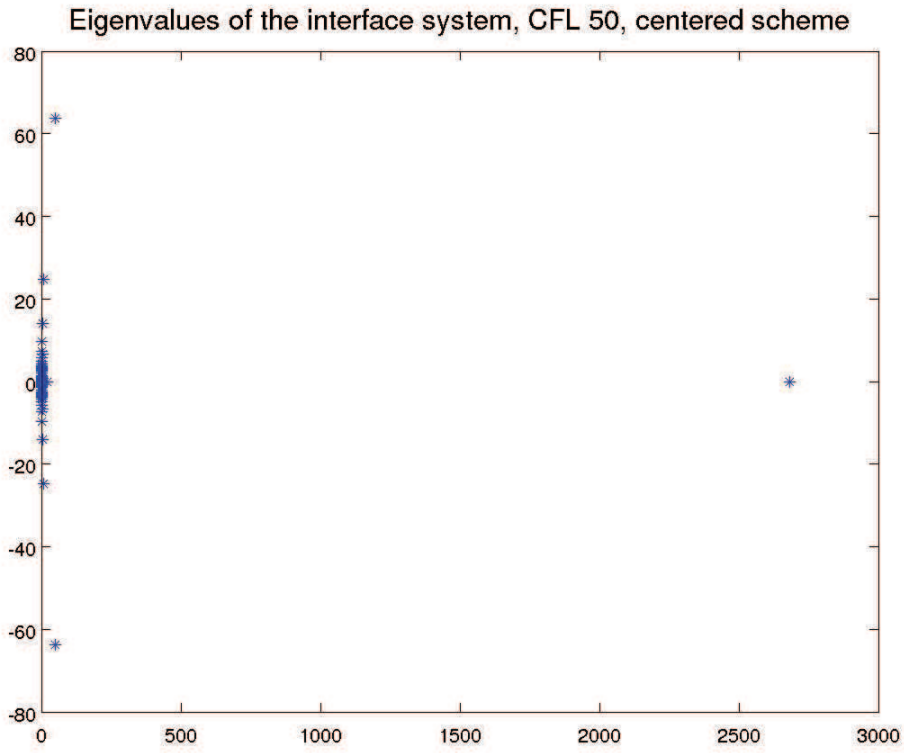

FIGURE 5.31 – *CFL* 50, 40×40 *cells*, 2 *processors*

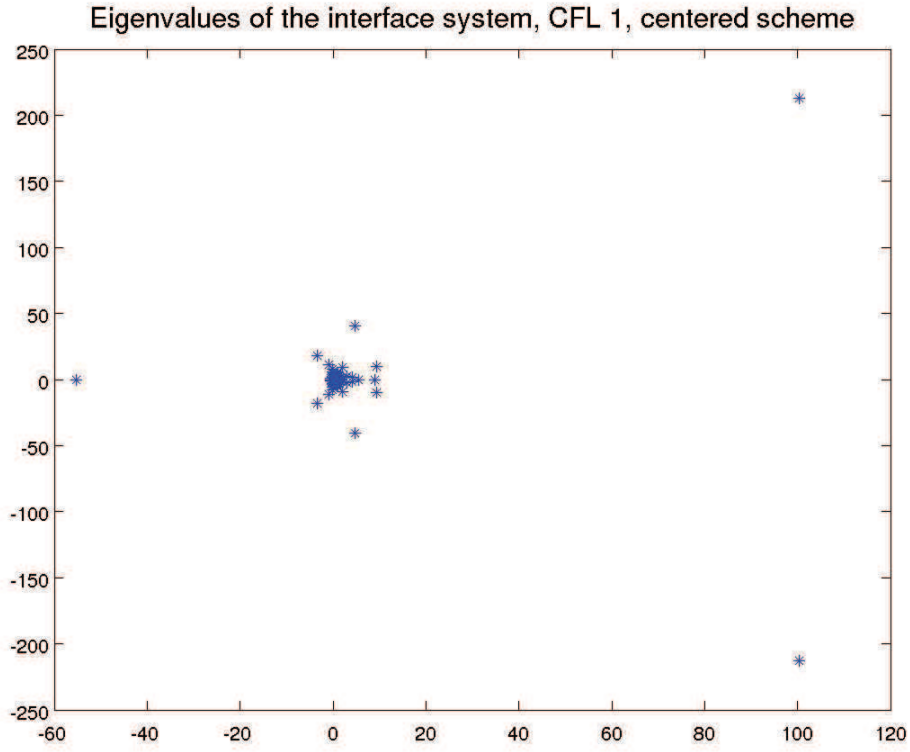

FIGURE 5.32 – *CFL* 1, 40×40 *cells*, 4 *processors*

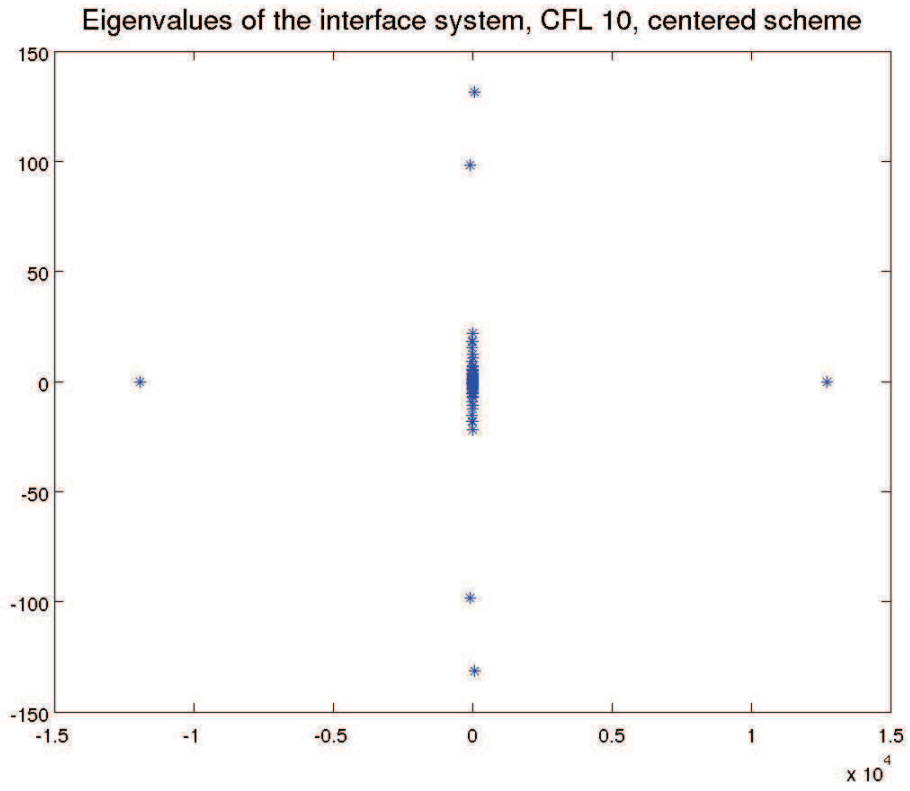

FIGURE 5.33 – *CFL* 10, 40×40 *cells*, 4 *processors*

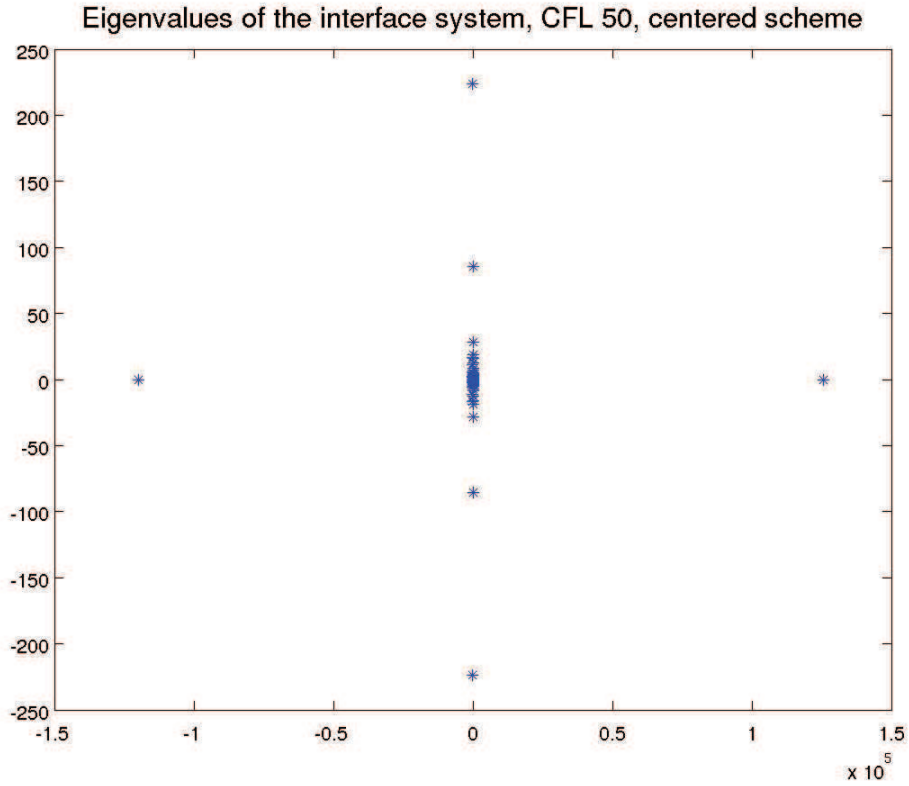

FIGURE 5.34 – *CFL* 50, 40×40 *cells*, 4 *processors*

on four processors using the centered scheme with CFL = 1, 10 and 50 respectively. We can see that like the case of the upwind scheme, there are some isolated eigenvalues. Then if we could eliminate those eigenvalues, we would obtain a faster solver.

Conclusion générale

Ce travail a été consacré à la simulation numérique des équations de la mécanique de fluides par des méthodes de volumes finis implicites.

Tout d'abord, nous avons étudié et mis en place une version implicite du schéma de Roe pour les écoulements monophasiques compressibles. Grâce à la méthode de Newton pour résoudre les systèmes nonlinéaires, nos schémas sont conservatifs.
Les schémas de volumes finis décentrés habituels sont peu satisfaisants lorsque le nombre de Mach devient faible (Guillard and Viozat (1999), Guillard and Murrone (2004) et Dellacherie (2010)). La solution calculée dépend fortement du nombre de Mach et du maillage. Ainsi, il est pratiquement impossible de calculer un écoulement multidimensionel à Mach 10^{-2} sur un maillage "raisonable". A cette fin, on a proposé d'utiliser le schéma centré implicite Dao et al. (2011b) (qui est inconditionellement stable et d'ordre 2). Les essais numériques effectués dans cette thèse ont montré que ce schéma peux capturer la bonne solution. Malheureusement, pour les schémas implicites il faut résoudre un système nonlinéaire à chaque pas de temps. Cette résolution peut représenter plus de 80% du temps de calcul total du code. Il est donc impératif d'utiliser des algorithmes de résolution performants. Pour des matrices de taille grande, on utilise souvent des méthodes itératives dont la convergence dépend de leur spectre. Nous avons donc étudié le spectre du système linéaire et proposé la stratégie de Scaling pour améliorer le conditionnement de la matrice. Cette stratégie est une transformation similaire de la matrice (en utilisant son spectre) pour que les coefficients en dehors de la diagonale aient le même d'ordre de grandeur. Combinée avec le préconditionneur classique ILU, notre stratégie de Scaling a réduit de façon significative le nombre d'itération GMRES du système local et le temps de calcul. Cette stratégie a fait l'objet d'une publication Dao et al. (2011b).

Nous avons ensuite étudié et implémenté la méthode de décomposition de domaine pour les écoulements monophasiques (équations d'Euler, équations de Navier-Stokes compressibles). Nous avons proposé une nouvelle variable interface inspirée par l'approche présentée dans Dolean and Lanteri (2001). Notre variable interface rend la méthode du complément de Schur plus facile à construire que celle utilisée dans Dolean and Lanteri (2001). De plus, elle permet de traiter les termes de diffusion. Nos schémas sont conservatifs grâce à la méthode de Newton utilisée pour résoudre les systèmes non linéaires. L'utilisation du solveur itératif GMRES plutôt que Richardson pour le système interface apporte aussi une amélioration des performances par rapport à Dolean and Lanteri (2001). Nous pouvons également découper notre domaine de calcul en un nombre quelconque

de sous-domaines.
En utilisant la stratégie de Scaling pour le système interface, nous avons amélioré le conditionnement de la matrice et à réduire le nombre d'itérations GMRES de ce système.
Une publlication Dao et al. (2011a) est consacrée à ce travail.

Dans un second temps, notre travail de thèse a porté sur la simulation numérique des écoulements diphasiques. Nous avons étudié et implémenté les schémas numériques de type Roe pour le modèle bi-fluide. Nous avons également proposé une correction entropique pour rendre le schéma de Roe positif. Des tests classiques (comme le tube à choc, la sédimentation etc ..) ont été réalisés pour montrer le bon comportement de ce schéma.

Nous avons également étudié le comportement asymptotique du modèle bi-fluide dans le cas où les deux phases sont supposées incompressibles. Cela donne un système de deux équations aux dérivées partielles du premiers ordre dont il est plus simple à étudier l'hyperbolicité ainsi que le comportement lorque un des phases disparaît. Ce travail donne suite à un stage (en ce moment) et une thèse qui commencera l'année prochaine.

Nous avons ensuite appliqué la méthode de décomposition de domaine au modèle bi-fluide et étudié son comportement dans le cas diphasique. En comparaison avec le calcul distribué classique, nous avons montré que notre méthode est plus robust et efficace.
Suite au succès de la stratégie de Scaling pour les écoulements monophasiques, cette stratégie a été appliquée au cas diphasique et elle a amélioré les performances.
Ce travail a donné lieu aux publications Dao et al. (2012a) et Dao et al. (2012b).

Les méthodes de décomposition de domaine que nous avons mises au point pourront être, par la suite, utilisées dans des logiciels scientifiques simulant les écoulements dans le circuit primaire d'un réacteur nucléaire à eau sous pression comme FLICA-4 (Toumi et al. (2000) ou FLICA-OVAP (Fillion et al. (2009)). Grâce à l'utilisation de l'interface ICoCo (Perdu (2006)) et la nouvelle variable interface, on peut facilement appliquer la méthode à d'autres modèles écoulements avec la possiblité de changer de loi d'état, de géometrie. On peut aussi appliquer la méthode aux problèmes de couplage entre les écoulements monophasique et diphasique.

On peut aussi améliorer la méthode de decomposition de domaine en cherchant des nouveaux préconditionneurs pour le système interface. On peut également chercher des nouvelle variable interface pour améliorer la condition de tranmission entre les subdomaines voisins.

Annexe

6

6.1 Primitive Variables and Its Computation from Conservative Variables

Primitive Aariables To obtain the quasi-linear form of the two-fluid model, we have defined the vector of conservative variables

$$\mathbf{U} =^t \left(m_g,\, q_g,\, m_l,\, q_l\right). \tag{6.1}$$

however, the variables which is used in the Jacobian matrix and its spectrum are called primitive variables

$$\mathbf{V} =^t \left(\alpha_g,\, p,\, u_g,\, u_l\right), \tag{6.2}$$

Here, we are interested in the computation of $\mathbf{V}$ from $\mathbf{U}$.
When computing u_k from the conservative variables m_k, q_k by the formula $u_k = \frac{q_k}{m_k}$, we have a singularity of the type $\frac{0}{0}$ when phase k is absent.
The computation of α_g and p is more complicated because we have to solve the equation

$$f\begin{pmatrix} \alpha_g \\ p \end{pmatrix} = \begin{pmatrix} m_g \\ m_l \end{pmatrix}. \tag{6.3}$$

Before solving this equation, we need to prove that it has solutions. The Jacobian matrix of f and its determinant are determined from the system (3.26) :

$$\nabla f = \begin{pmatrix} \rho_g & \frac{\alpha_g}{c_g^2} \\ -\rho_l & \frac{\alpha_l}{c_l^2} \end{pmatrix}, \qquad |\nabla f| = \gamma^{-2}.$$

For physically acceptable values we have : $\gamma^2 > 0$ and $|\nabla f| \neq 0$. Then the matrix ∇f is always inversible. Therefore f is locally invertible and its inverse has Jacobian matrix :

$$(\nabla f)^{-1} = \gamma^2 \begin{pmatrix} \frac{\alpha_l}{c_l^2} & -\frac{\alpha_g}{c_g^2} \\ \rho_l & \rho_g \end{pmatrix}.$$

Newton Method for α and p We can solve the system (6.3) by Newton iterative method :

$$V^{k+1} = V^k - (\nabla f)^{-1}_{V^k}(f(V^k) - f_{obj}), \tag{6.4}$$

where V^k denotes the current approximate vector ${}^t(\alpha_g, p)$ and $f_{obj} = {}^t(m_g, m_l)$ is the objective value of f.
By detailing the equation of Newton scheme (6.4) we obtain

$$\alpha_g{}^{k+1} = \alpha_g{}^k - \gamma^2 \left[\frac{\alpha_l}{c_l^2}\left(\alpha_g\rho_g - m_g\right) - \frac{\alpha_g}{c_g^2}\left(\alpha_l\rho_l - m_l\right)\right]^k \tag{6.5}$$

$$p^{k+1} = p^k - \gamma^2 \left[\rho_l\left(\alpha_g\rho_g - m_g\right) + \rho_g\left(\alpha_l\rho_l - m_l\right)\right]^k. \tag{6.6}$$

It is well known that the Newton scheme convergence is quadratic if the initial guess is close to the exact solution. In the general case, the Newton method may have problem of oscillation or even divergence, and we can not assure that the inegality $0 \leq \alpha_g{}^{k+1} \leq 1$ will be satisfied. When $\alpha_g{}^{k+1} \leq 0$, it is better to use a different method, such as bisection for $\alpha_g{}^{k+1}$. By denoting

$$\begin{aligned} \Delta\alpha_g{}^{Newton} &= -\gamma^2 \left[\frac{\alpha_l}{c_l^2}\left(\alpha_g\rho_g - m_g\right) - \frac{\alpha_g}{c_g^2}\left(\alpha_l\rho_l - m_l\right)\right]^k, \\ \beta &= \frac{9}{10}\frac{\alpha_g}{\Delta\alpha_g{}^{Newton}}, \end{aligned}$$

the algorithm consists in dividing the value of α_g by 10 can be written as a relaxed Newton method :

$$V^{k+1} = V^k - \beta(\nabla f)^{-1}_{V^k}(f(V^k) - f_{obj}), \tag{6.7}$$

Condition number of the function f The solution of $f = 0$ is a ill-conditioned problem. To prove that, we compute the condition number of ∇f :

$$\begin{aligned} (\nabla f)^t(\nabla f) &= \begin{pmatrix} \rho_g{}^2 + \frac{\alpha_g{}^2}{c_g^4} & \frac{\alpha_g\alpha_l}{c_g^2c_l^2} - \rho_g\rho_l \\ \frac{\alpha_g\alpha_l}{c_g^2c_l^2} - \rho_g\rho_l & \rho_l{}^2 + \frac{\alpha_l{}^2}{c_l^4} \end{pmatrix}, \\ \text{trace}(\nabla f^t\nabla f) &= \rho_g{}^2 + \rho_l{}^2 + \frac{\alpha_g{}^2}{c_g^4} + \frac{\alpha_l{}^2}{c_l^4}, \\ \det(\nabla f^t\nabla f) &= \left(\frac{\alpha_g\rho_l}{c_g^2} + \frac{\alpha_l\rho_g}{c_l^2}\right)^2, \\ \Delta(\nabla f^t\nabla f) &= \left((\rho_g + \frac{\alpha_l}{c_l^2})^2 + (\rho_l + \frac{\alpha_g}{c_g^2})^2\right)\left((\rho_g - \frac{\alpha_l}{c_l^2})^2 + (\rho_l - \frac{\alpha_g}{c_g^2})^2\right), \\ \lambda_\pm &= \frac{1}{2}(\text{trace} \pm \sqrt{\Delta}) \end{aligned}$$

Considering that $\frac{\alpha_l}{c_l^2} \ll \rho_g$ and $\frac{\alpha_g}{c_g^2} \ll \rho_l$ we find the limit values $\lambda_+ \approx \frac{1}{2}({\rho_g}^2 + {\rho_l}^2)$ and $\lambda_- \approx \frac{1}{2}(\frac{{\alpha_g}^2}{c_g^4} + \frac{{\alpha_l}^2}{c_l^4})$. Hence we have

$$\kappa(\nabla f) = \sqrt{\frac{{\rho_g}^2 + {\rho_l}^2}{\frac{{\alpha_g}^2}{c_g^4} + \frac{{\alpha_l}^2}{c_l^4}}} \approx \frac{\rho_l c_g^2}{\alpha_g}.$$

For typical air water mixture at a pressure of 1 bar ($\alpha_g = \alpha_l = 0.5$, $\rho_g = 1$, $\rho_l = 1000$, $c_g = 330$, $c_l = 1500$),

$$\kappa(\nabla f) = 2.\,10^8,$$

which makes the Newton scheme poorly reliable.

In the limit $\alpha_g = 0$ While the Newton method (6.5-6.6) does not assure that $0 \leq {\alpha_g}^{k+1} \leq 1$, in the case $m_g = 0$, we directly have ${\alpha_g}^k = 0$ and we need only perform the second iteration

$$p^{k+1} = p^k - c_l(p^k)^2 \left(\rho_l(p^k) - m_l\right),$$

which converges in practice.

6.2 Jacobian Matrix of the Incompressible Limit of the Two-phase Flow

In this section, we want to give the Jacobian matrix of the incompressible limit of the two-phase flow. We recall that the flux of the incompressible system is given as :

$$F = \begin{pmatrix} \frac{K[\beta(\rho_g+\rho_l)-\rho_g\rho_l]}{\beta} + \frac{(\beta-\rho_g)(\beta-\rho_l)\omega}{\beta(\rho_g-\rho_l)} \\ \\ \frac{(\beta^2-\rho_g\rho_l)\omega^2}{2\beta^2(\rho_g-\rho_l)} + \frac{K\rho_g\rho_l\omega}{\beta^2} - \frac{K^2\rho_g\rho_l(\rho_g-\rho_l)}{2\beta^2} - \frac{\delta p(\beta-\rho_g)}{\rho_g-\rho_l} \end{pmatrix}$$

Then, from (3.61) and (3.62) we have :

$$\partial_\beta \alpha_g = -\frac{1}{\rho_g - \rho_l}, \quad \partial_\omega \alpha_g = 0 \tag{6.8}$$

$$\partial_\beta \alpha_l = \frac{1}{\rho_g - \rho_l}, \quad \partial_\omega \alpha_l = 0 \tag{6.9}$$

$$\partial_\beta u_g = \frac{\rho_l(\frac{\omega}{\rho_g-\rho_l} - K)}{\beta^2} = \frac{\rho_l(u_g - u_l)}{\beta(\rho_g - \rho_l)}, \quad \partial_\omega u_g = \frac{\alpha_l}{\beta} \tag{6.10}$$

$$\partial_\beta u_l = \frac{\rho_g(\frac{\omega}{\rho_g-\rho_l} - K)}{\beta^2} = \frac{\rho_g(u_g - u_l)}{\beta(\rho_g - \rho_l)}, \quad \partial_\omega u_l = -\frac{\alpha_g}{\beta} \tag{6.11}$$

Jacobian matrix :

$$A = \begin{pmatrix} \frac{\partial F_1}{\partial \beta} & \frac{\partial F_1}{\partial \omega} \\ \\ \frac{\partial F_2}{\partial \beta} & \frac{\partial F_2}{\partial \omega} \end{pmatrix}$$

$$
\begin{aligned}
\frac{\partial F_1}{\partial \beta} &= \rho_l u_g \frac{\partial \alpha_g}{\partial \beta} + \rho_l \alpha_g \frac{\partial u_g}{\partial \beta} + \rho_g u_l \frac{\partial \alpha_l}{\partial \beta} + \rho_g \alpha_l \frac{\partial u_l}{\partial \beta} \\
&= \frac{1}{\rho_g - \rho_l}\{\rho_g u_l - \rho_l u_g + \frac{(u_g - u_l)}{\beta}(\alpha_g \rho_l^2 + \alpha_l \rho_g^2)\} \\
&= \frac{1}{\beta(\rho_g - \rho_l)}\{(\rho_g u_l - \rho_l u_g)(\alpha_g \rho_l + \alpha_l \rho_g) + u_g - u_l)(\alpha_g \rho_l^2 + \alpha_l \rho_g^2)\} \\
&= \frac{\alpha_g \rho_l u_l + \alpha_l \rho_g u_g}{\beta} \\
\frac{\partial F_1}{\partial \omega} &= \rho_l \alpha_g \frac{\partial u_g}{\partial \omega} + \rho_g \alpha_l \frac{\partial u_l}{\partial \omega} = -\frac{\alpha_g \alpha_l (\rho_g - \rho_l)}{\beta} \\
\frac{\partial F_2}{\partial \beta} &= \rho_g u_g \frac{\partial u_g}{\partial \beta} - \rho_l u_l \frac{\partial u_l}{\partial \beta} + \delta p \frac{\partial \alpha_g}{\partial \beta} \\
&= \frac{\rho_g \rho_l u_g (u_g - u_l)}{\beta(\rho_g - \rho_l)} - \frac{\rho_g \rho_l u_l (u_g - u_l)}{\beta(\rho_g - \rho_l)} - \frac{\delta p}{\rho_g - \rho_l} \\
&= \frac{(u_g - u_l)^2 \rho_g \rho_l}{\beta(\rho_g - \rho_l)} - \frac{\delta p}{\rho_g - \rho_l} \\
\frac{\partial F_2}{\partial \omega} &= \rho_g u_g \frac{\partial u_g}{\partial \omega} - \rho_l u_l \frac{\partial u_l}{\partial \omega} = \frac{\alpha_g \rho_l u_l + \alpha_l \rho_g u_g}{\beta}
\end{aligned}
$$

Then, we have the Jacobian matrix :

$$
A = \begin{pmatrix}
\frac{\alpha_g \rho_l u_l + \alpha_l \rho_g u_g}{\alpha_g \rho_l + \alpha_l \rho_g} & -\frac{\alpha_g \alpha_l (\rho_g - \rho_l)}{\alpha_g \rho_l + \alpha_l \rho_g} \\
\frac{(u_g - u_l)^2 \rho_g \rho_l}{(\alpha_g \rho_l + \alpha_l \rho_g)(\rho_g - \rho_l)} - \frac{\delta p}{\rho_g - \rho_l} & \frac{\alpha_g \rho_l u_l + \alpha_l \rho_g u_g}{\alpha_g \rho_l + \alpha_l \rho_g}
\end{pmatrix}
$$

Or we can write A as follows :

$$
A = \begin{pmatrix}
\frac{1}{\beta^2}\left[K\rho_g\rho_l + \frac{\omega(\beta^2 - \rho_g\rho_l)}{\rho_g - \rho_l}\right] & \frac{(\beta - \rho_g)(\beta - \rho_l)}{\beta(\rho_g - \rho_l)} \\
\frac{[\omega - K(\rho_g - \rho_l)]^2 \rho_g \rho_l}{\beta^3(\rho_g - \rho_l)} - \frac{\delta p}{\rho_g - \rho_l} & \frac{1}{\beta^2}\left[K\rho_g\rho_l + \frac{\omega(\beta^2 - \rho_g\rho_l)}{\rho_g - \rho_l}\right]
\end{pmatrix}
$$

$$
\begin{aligned}
\frac{\partial \lambda_1}{\partial \beta} &= \frac{-2K\rho_g\rho_l}{\beta^3} + \frac{2\omega\rho_g\rho_l}{(\rho_g - \rho_l)\beta^3} \\
&+ \frac{-\frac{\Delta p}{\beta^2} - \frac{\rho_g\rho_l}{\beta^5}\left[(\beta - \rho_g)(-\beta + 2\rho_l) + (\beta - \rho_l)(-\beta + 2\rho_g)\right](\frac{\omega}{\rho_g - \rho_l} - K)^2}{2\sqrt{\frac{1}{\beta}\left(\Delta p + \frac{(\beta - \rho_g)(\beta - \rho_l)\rho_g\rho_l}{\beta^3}(\frac{\omega}{\rho_g - \rho_l} - K)^2\right)}} \\
\frac{\partial \lambda_1}{\partial \omega} &= \frac{\beta^2 - \rho_g\rho_l}{\beta^2(\rho_g - \rho_l)} + \frac{(\beta - \rho_g)(\beta - \rho_l)\rho_g\rho_l(\frac{\omega}{\rho_g - \rho_l} - K)}{\beta^4(\rho_g - \rho_l)\sqrt{\frac{1}{\beta}\left(\Delta p + \frac{(\beta - \rho_g)(\beta - \rho_l)\rho_g\rho_l}{\beta^3}(\frac{\omega}{\rho_g - \rho_l} - K)^2\right)}}
\end{aligned}
$$

6.3 Spectrum of the Jacobian Matrix

From 3.64, we obtain the following eigenvectors of the Jacobian matrix :

$$r_1 = \begin{pmatrix} -\frac{\alpha_g\alpha_l(\rho_g-\rho_l)}{\beta} \\ \sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}(u_g-u_l)^2\right)} \end{pmatrix} \tag{6.12}$$

$$r_2 = \begin{pmatrix} \frac{\alpha_g\alpha_l(\rho_g-\rho_l)}{\beta} \\ \sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}(u_g-u_l)^2\right)} \end{pmatrix} \tag{6.13}$$

These eigenvectors can be written as :

$$r_1 = \begin{pmatrix} \frac{(\beta-\rho_g)(\beta-\rho_l)}{\beta(\rho_g-\rho_l)} \\ \sqrt{\frac{1}{\beta}\left(\Delta p + \frac{(\beta-\rho_g)(\beta-\rho_l)\rho_g\rho_l}{\beta^3}(\frac{\omega}{\rho_g-\rho_l}-K)^2\right)} \end{pmatrix} \tag{6.14}$$

$$r_2 = \begin{pmatrix} -\frac{(\beta-\rho_g)(\beta-\rho_l)}{\beta(\rho_g-\rho_l)} \\ \sqrt{\frac{1}{\beta}\left(\Delta p + \frac{(\beta-\rho_g)(\beta-\rho_l)\rho_g\rho_l}{\beta^3}(\frac{\omega}{\rho_g-\rho_l}-K)^2\right)} \end{pmatrix} \tag{6.15}$$

From 3.63, the eigenvalues of A are given by :

$$\lambda_{1,2} = \frac{\alpha_g\rho_l u_l + \alpha_l\rho_g u_g}{\beta} \pm \sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}(u_g-u_l)^2\right)x} \tag{6.16}$$

$$= \frac{1}{\beta^2}\left[K\rho_g\rho_l + \frac{\omega(\beta^2-\rho_g\rho_l)}{\rho_g-\rho_l}\right] \pm \sqrt{\frac{1}{\beta}\left(\Delta p + \frac{(\beta-\rho_g)(\beta-\rho_l)\rho_g\rho_l}{\beta^3}(\frac{\omega}{\rho_g-\rho_l}-K)^2\right)}$$

Then, we have

$$\frac{\partial \lambda_1}{\partial \beta} = \frac{2\rho_g\rho_l(u_g-u_l)}{\beta^2(\rho_g-\rho_l)} - \frac{\frac{\Delta p}{\beta^2} + \frac{(u_g-u_l)^2\rho_g\rho_l\left(\alpha_g\rho_l-\alpha_l\rho_g-2\alpha_g\alpha_l(\rho_g-\rho_l)\right)}{2\beta^3(\rho_g-\rho_l)}}{2\sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}(u_g-u_l)^2\right)}}$$

$$\frac{\partial \lambda_2}{\partial \beta} = \frac{2\rho_g\rho_l(u_g-u_l)}{\beta^2(\rho_g-\rho_l)} + \frac{\frac{\Delta p}{\beta^2} + \frac{(u_g-u_l)^2\rho_g\rho_l(\alpha_g\rho_l-\alpha_l\rho_g-2\alpha_g\alpha_l(\rho_g-\rho_l))}{2\beta^3(\rho_g-\rho_l)}}{2\sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}(u_g-u_l)^2\right)}}$$

$$\frac{\partial \lambda_1}{\partial \omega} = \frac{\alpha_l^2\rho_g-\alpha_g^2\rho_l}{\beta^2} - \frac{\alpha_g\alpha_l\rho_g\rho_l(u_g-u_l)}{\beta^3\sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}(u_g-u_l)^2\right)}}$$

$$\frac{\partial \lambda_2}{\partial \omega} = \frac{\alpha_l^2\rho_g-\alpha_g^2\rho_l}{\beta^2} + \frac{\alpha_g\alpha_l\rho_g\rho_l(u_g-u_l)}{\beta^3\sqrt{\frac{1}{\alpha_g\rho_l+\alpha_l\rho_g}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\alpha_g\rho_l+\alpha_l\rho_g}(u_g-u_l)^2\right)}}$$

Thus, we have :

$$\begin{aligned}\nabla\lambda_1\cdot r_1 &= -\frac{3\alpha_g\alpha_l\rho_g\rho_l(u_g-u_l)}{\beta^3} + \frac{\frac{\Delta p\alpha_g\alpha_l(\rho_g-\rho_l)}{\beta^3} + \frac{(u_g-u_l)^2\rho_g\rho_l\alpha_g\alpha_l\left(\alpha_g\rho_l-\alpha_l\rho_g-2\alpha_g\alpha_l(\rho_g-\rho_l)\right)}{2\beta^4}}{2\sqrt{\frac{1}{\beta}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\beta}(u_g-u_l)^2\right)}} \\ &+ \frac{\alpha_l^2\rho_g-\alpha_g^2\rho_l}{\beta^2}\sqrt{\frac{1}{\beta}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\beta}(u_g-u_l)^2\right)} \\ \nabla\lambda_2\cdot r_2 &= \frac{3\alpha_g\alpha_l\rho_g\rho_l(u_g-u_l)}{\beta^3} + \frac{\frac{\Delta p\alpha_g\alpha_l(\rho_g-\rho_l)}{\beta^3} + \frac{(u_g-u_l)^2\rho_g\rho_l\alpha_g\alpha_l\left(\alpha_g\rho_l-\alpha_l\rho_g-2\alpha_g\alpha_l(\rho_g-\rho_l)\right)}{2\beta^4}}{2\sqrt{\frac{1}{\beta}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\beta}(u_g-u_l)^2\right)}} \\ &+ \frac{\alpha_l^2\rho_g-\alpha_g^2\rho_l}{\beta^2}\sqrt{\frac{1}{\beta}\left(\Delta p - \frac{\alpha_g\alpha_l\rho_g\rho_l}{\beta}(u_g-u_l)^2\right)}\end{aligned}$$

Bibliographie

A.Harten. High resolution schemes for hyperbolic conservation laws. *J. Comput. Phys.*, 49, 1983.

A.Harten. On a Class of High Resolution Total-Variation-Stable Finite-Difference Schemes. *SIAM J. Numer. Anal.*, 21, 1984.

B. F. Armaly, F. Durst, J. C. F. Pereira, and B. Schoung. Experimental and theorical investigation of backward-facing step flow. *J. Fluid Mech.*, 127, 1983.

W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9(1), 1951.

Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2012. http ://www.mcs.anl.gov/petsc.

Michele Benzi. Preconditioning Techniques for Large Linear Systems : A Survey. *J. Comput. Phys.*, 182, 2002.

V. Bergeaud, P. Fillion, and J. Dérouillat. Etude bibliographique sur l'inversion parallèle des matrices ovap. Technical report, Rapport CS/311-1/AB06A002-010/RAP/07-065 version 1.0, 2007.

J.-F. Bourgat, R. Glowinski, P. Le Tallec, and M. Vidrascu. Variational formulation and algorithm for trace operator in domain decomposition calculations. In T.F. Chan, R. Glowinski, J. Périaux, O.B. Widlund editors. In *Domain Decomposition Methods*, 1989.

X. C. Cai and Y. Saad. Overlapping domain decomposition algorithms for general sparse matrices. *Numer. Lin. Alg. Applic.*, 3, 1996.

X. C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparselinear systems. *SIAM Journal on Scientific Computing*, 21, 1999.

Philippe Chevalier and Frédéric Nataf. Symmetrized method with optimized second-order conditions for the Helmholtz equation. In *Domain decomposition methods, 10 (Boulder, CO, 1997)*, pages 400–407. Amer. Math. Soc., Providence, RI, 1998.

F. Coquel, K. El Amine, E. Godlewski, B. Perthame, and P. Rascle. A Numerical Method Using Upwind Schemes for the Resolution of Two-Phase Flows. *J. Comput. Phys.*, 136, 1997.

T.H. Dao, M. Ndjinga, and F. Magoulès. A schur complement method for compressible Navier-Stokes equations. In *Proceedings of the 20th International Conference on Domain Decomposition Methods*, 2011a. In press.

T.H. Dao, M. Ndjinga, and F. Magoulès. Comparison of Upwind and Centered Schemes for Low Mach Number Flows. In *Proceedings of the International Symposium Finite Volumes for Complex Application VI*. Springer Proceedings in Mathematics 4, 2011b.

T.H. Dao, M. Ndjinga, and F. Magoulès. A Schur Complement Method for Compressible Two-Phase Flow Models. In *Proceedings of the 21th International Conference on Domain Decomposition Methods*, 2012a. Submitted.

T.H. Dao, M. Ndjinga, and F. Magoulès. A Schur complement method for compressible flows, 2012b. In preparation.

S. Dellacherie. Analysis of Godunov type schemes applied to the compressible Euler system at low Mach number. *J. Comput. Phys.*, 229, 2010.

B. Després and François Dubois. *Systèmes hyperboliques de lois de conservations*. Les Editions de l'Ecole Polytechnique, 2005.

Bruno Després. Domain decomposition method and the Helmholtz problem.II. In *Second International Conference on Mathematical and Numerical Aspects of Wave Propagation (Newark, DE, 1993)*, pages 197–206, Philadelphia, PA, 1993. SIAM.

Bruno Després, Patrick Joly, and Jean E. Roberts. A domain decomposition method for harmonic Maxwell equations. In *Iterative methods in linear algebra*, pages 475–484, Amsterdam, 1992. North-Holland.

V. Dolean and S. Lanteri. A domain decomposition approach to finite volume solution of the Euler equations on unstructured triangular meshes. *Int. J. Numer. Meth. Fluids*, 37(6), 2001.

D. A. Drew and S. L. Passman. *Theory of Multicomponents Fluids*. Springer-Verlag, New York, 1999.

I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, London, 1986.

S. Evje and T. Flatten. Hybrid flux-splitting schemes for a common two-fluid model. *J. Comput. Phys.*, 192, 2003.

C. Farhat and F. X. Roux. *Implicit Parallel Processing in Structural Mechanics*, volume 1–124. North Holland, 1994.

P. Fillion, A. Chanoine, S. Dellacherie, and A. Kumbaro. FLICA-OVAP : a New Platform for Core Thermal-hydraulic Studies. In *NURETH-13*, 2009.

R. Fletcher. Conjugate gradient methods for indefinite systems. *Lecture Notes in Mathematics*, 506, 1976.

R. W. Freund and N. M. Nachtigal. QRM : a quasi-minimal residual method for non-Hermitian linear systems. *Numeriske Mathematik*, 60, 1991.

Martin J. Gander, Frédéric Magoulès, and Frédéric Nataf. Optimized Schwarz methods without overlap for the Helmholtz equation. *SIAM J. Sci. Comput.*, 24(1) :38–60, jan 2002. ISSN 1064-8275.

Martin J. Gander, Laurence Halpern, and Frédéric Magoulès. An optimized Schwarz method with two-sided Robin transmission conditions for the Helmholtz equation. *Int. J. for Num. Meth. in Fluids*, 55(2) :163–175, 2007.

U. Ghia, K.N. Ghia, and C.T. Shin. High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method. *J. Comput. Phys.*, 48, 1982.

E. Godlewski and P.A. Raviart. *Numerical Approximation of Hyperbolic Systems of Conservation Laws*. Springer Verlag, 1996.

H. Guillard and A. Murrone. On the behaviour of upwind schemes in the low Mach number limit : II. Godunov type schemes. *Computers & Fluids*, 33, 2004.

H. Guillard and C. Viozat. On the behaviour of upwind schemes in the low Mach number limit. *Computers & Fluids*, 28, 1999.

M. Hestenes and E. Steifel. Method of conjugate gradients for solving linear systems. *Journal of Research*, 20, 1952.

M. Ishii. *Thermo-fluid Dynamic Theory of Two-phase Flow*. Eyrolles, Paris, 1975.

Caroline Japhet, Frédéric Nataf, and Francois Rogier. The optimized order 2 method. Application to convection-diffusion problems. *Future Generation Computer Systems*, 18(1) :17–30, 2001.

D. E. Keyes. How scalable is domain decomposition in practice ? In *Proceedings of the 11th International Conference on Domain Decomposition Methods*, 1998.

B. L. Keyfitz, R. Sanders, and M. Sever. Lack of hyperbolicity in the two-fluid model for two-phase incompressible flow. *Discrete and continuous dynamical systems-Series B*, 3, 2003.

P. G. LeFloch. *Hyperbolic Systems of Conservation Laws*. Springer, 2002.

R.J. LeVeque. *Numerical Methods for Conservation Laws*. Springer, 1992.

R.J. LeVeque. *Finite-Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

Pierre-Louis Lions. On the Schwarz alternating method. II. In Tony Chan, Roland Glowinski, Jacques Périaux, and Olof Widlund, editors, *Domain Decomposition Methods*, pages 47–70, Philadelphia, PA, 1989. SIAM.

Pierre-Louis Lions. On the Schwarz alternating method. III : a variant for nonoverlapping subdomains. In Tony F. Chan, Roland Glowinski, Jacques Périaux, and Olof Widlund, editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations , held in Houston, Texas, March 20-22, 1989*, Philadelphia, PA, 1990. SIAM.

Y. Maday and F. Magoulès. Absorbing interface conditions for domain decomposition methods : a general presentation. *Computer Methods in Applied Mechanics and Engineering*, 195(29-32) :3880–3900, 2006.

F. Magoulès, P. Ivànyi, and B.H.V. Topping. Non-overlapping Schwarz methods with optimized transmission conditions for the Helmholtz equation. *Computer Methods in Applied Mechanics and Engineering*, 193(45-47) : 4797–4818, 2004a.

F. Magoulès, F.-X. Roux, and S. Salmon. Optimal discrete transmission conditions for a non-overlapping domain decomposition method for the Helmholtz equation. *SIAM Journal on Scientific Computing*, 25(5) :1497–1515, 2004b.

J. A. Meijerink and H. A. van der Vorst. An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix. *Math. Comp.*, 31, 1977.

S. T. Munkejord. *Analysis of the two-fluid model and the drift-flux model for numerical calculation of two-phase flow*. PhD thesis, Norwegian University of Science and Technology, 2005.

M. Ndjinga. *Quelques aspects d'analyse et de modélisation des systèmes issus des écoulements diphasiques*. PhD thesis, Ecole Centrale Paris, 2007a.

M. Ndjinga. Influence of interfacial pressure on the hyperbolicity of the two-fluid model. *C. R. Acad. Sci. Paris*, 344, 2007b.

M. Ndjinga. Spectral stability of finite volume schemes for linear hyperbolic systems. *C. R. Acad. Sci. Paris*, 349, 2011.

M. Ndjinga, A. Kumbaro, F. De Vuyst, and P. Laurent-Gengoux. Numerical simulation of hyperbolic two-phase flow models using a Roe-type solver. *Nucl. Eng. Design*, 238, 2008.

P. Okunev and C. R. Johnson. Necessary and sufficient conditions for existence of the lu factorization of an arbitrary matrix, 2005. http ://arxiv.org/abs/math.NA/0506382.

F. Perdu. Proposition d'une interface de couplage en vue des couplages de codes dans neptune version 1. Technical report, CEA, 2006. Note technique.

T.H. Pulliam. Development of implicit methods in CFD NASA Ames Research Center 1970s-1980s. *Computers Fluids*, 41, 2011.

A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, USA, 1999.

V. H. Ransom. Numerical Benchmark Tests. *Multiphase Science and Technology*, 3, 1987.

P.L Roe. Approximate Riemann solvers, parameter vectors and difference schemes. *J. Comput. Phys.*, 43, 1981.

F.-X. Roux, F. Magoulès, L. Series, and Y. Boubendir. Approximation of optimal interface boundary conditions for two-Lagrange multiplier FETI method. In *Proceedings of International Conference on Domain Decomposition Methods*. Springer Lecture Notes in Computational Science and Engineering, 2005.

Y. Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. Siam, 2000.

Y. Saad and M. H. Schultz. GMRES : A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3), 1986.

Y. Saad and B. Suchomel. ARMS : an algebraic recursive multilevel solver for general sparse linear systems. *Numer. Linear Algebra Appl.*, 5, 2002.

Y. Saad and J. Zhang. BILUM, Block versions of multi-elimination and multilevel ILU preconditioner for general sparse linear systems. *SIAM Journal on Scientific and Statistical Computing*, 20, 1999.

H. A. Schwarz. Uber einige Abbildungensaufgaben. *J. Reine Angew. Math.*, 70, 1869.

B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition : Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 10, 1989.

P. Le Tallec, Y.-H. De Roeck, and M. Vidrascu. Domain-decomposition methods for large linearly elliptic three dimensional problems. *J. of Computational and Applied Mathematics*, 34, 1991.

Eleuterio F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics : A Practical Introduction*. Springer, 2009.

A. Toselli and O. Widlund. *Domain Decomposition Methods-Algorithms and Theory*. Springer Verlag, 2005.

I. Toumi and A. Kumbaro. An Approximate Linearized Riemann Solver for a Two-Fluid Model. *J. Comput. Phys.*, 124, 1999.

I. Toumi, A. Kumbaro, and H. Paillère. Approximate riemann sovlers and flux vector splitting schemes for two phase flows. Technical report, CEA, 1999. Rapport CEA-R-5849.

I. Toumi, A. Bergeron, D. Gallo, E. Royer, and D. Caruge. FLICA-4 : a three-dimensional two-phase flow computer code with advanced numerical methods for nuclear applications. *Nuclear Engineering and Design*, 200, 2000.

L. N Trefethen and III D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.

H. van der Vorst. BI-CGSTAB : A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13, 1992.

R. Varga. *Matrix Iterative Analysis, Second Edition*. Springer-Verlag, Berlin, 2000.

R. S. Varga, E. B. Saff, and V. Mehrman. Incomplete factorizations of matrices and connections with H-matrices. *SIAM J. Numer. Anal.*, 17, 1980.

D. C. Wan, B. S. V. Patnaik, and G. W. Wei. Discrete Singular Convolution-Finite Subdomain Method for the Solution of Incompressible Viscous Flows. *J. Comput. Phys.*, 180, 2002.

J. Y. Zhu. The 2nd-order projection method for the backward-facing step flow. *J. Comput. Phys.*, 117, 1995.

Titre Simulation numérique d'écoulements diphasiques par décomposition de domaine

Résumé Ce travail a été consacré à la simulation numérique des équations de la mécanique des fluides par des méthodes de volumes finis implicites.
Tout d'abord, nous avons étudié et mis en place une version implicite du schéma de Roe pour les écoulements monophasiques et diphasiques compressibles. Grâce à la méthode de Newton utilisée pour résoudre les systèmes nonlinéaires, nos schémas sont conservatifs. Malheureusement, la résolution de ces systèmes est très coûteuse. Il est donc impératif d'utiliser des algorithmes de résolution performants. Pour des matrices de grande taille, on utilise souvent des méthodes itératives dont la convergence dépend de leur spectre. Nous avons donc étudié le spectre du système linéaire et proposé une stratégie de Scaling pour améliorer le conditionnement de la matrice. Combinée avec le préconditionneur classique ILU, notre stratégie de Scaling a réduit de façon significative le nombre d'itérations GMRES du système local et le temps de calcul. Nous avons également montré l'intérêt du schéma centré pour la simulation de certains écoulements à faible nombre de Mach.
Nous avons ensuite étudié et implémenté la méthode de décompostion de domaine pour les écoulements compressibles. Nous avons proposé une nouvelle variable interface qui rend la méthode du complément de Schur plus facile à construire et nous permet de traiter les termes de diffusion. L'utilisation du solveur itératif GMRES plutôt que Richardson pour le système interface apporte aussi une amélioration des performances par rapport aux autres méthodes. Nous pouvons également découper notre domaine de calcul en un nombre quelconque de sous-domaines. En utilisant la stratégie de Scaling pour le système interface, nous avons amélioré le conditionnement de la matrice et réduit le nombre d'itérations GMRES de ce système. En comparaison avec le calcul distribué classique, nous avons montré que notre méthode est robuste et efficace.

Mots-clés Equations d'Euler - Equations de Navier-Stokes - Modèle bifluide - Volumes finis - Schéma de Roe - Schéma implicite - Méthode de Newton - Préconditionneur - Décomposition de domaine - Complément de Schur - Calcul parallèle

Title Simulation of two-phase flows by domain decomposition

Abstract This thesis deals with numerical simulations of compressible fluid flows by implicit finite volume methods.
Firstly, we studied and implemented an implicit version of the Roe scheme for compressible single-phase and two-phase flows. Thanks to Newton method for solving nonlinear systems, our schemes are conservative. Unfortunately, the resolution of nonlinear systems is very expensive. It is therefore essential to use an efficient algorithm to solve these systems. For large size matrices, we often use iterative methods whose convergence depends on the spectrum. We have studied the spectrum of the linear system and proposed a strategy, called Scaling, to improve the condition number of the matrix. Combined with the classical ILU preconditioner, our strategy has reduced significantly the GMRES iterations for local systems and the computation time. We also show some satisfactory results for low Mach-number flows using the implicit centered scheme.
We then studied and implemented a domain decomposition method for compressible fluid flows. We have proposed a new interface variable which makes the Schur complement method easy to build and allows us to treat diffusion terms. Using GMRES iterative solver rather than Richardson for the interface system also provides a better performance compared to other methods. We can also decompose the computational domain into any number of subdomains. Moreover, the Scaling strategy for the interface system has improved the condition number of the matrix and reduced the number of GMRES iterations. In comparison with the classical distributed computing, we have shown that our method is more robust and efficient.

Keywords Euler equations - Navier-Stokes equations - Two-fluid model - Finite volume methods- Roe scheme - Implicit scheme - Newton method - Preconditioner - Domain decomposition - Schur complement - Parallel computing