Universidade Federal do Espírito Santo
Centro Tecnológico
Programa de Pós-Graduação em Informática

Sérgio Souza Bento

Nonlinear multiscale viscosity methods and time integration schemes for solving compressible Euler equations

Vitória - ES, Brasil

2018

Nonlinear Multiscale Viscosity Methods and Time Integration Schemes for Solving Compressible Euler Equations

Sérgio Souza Bento

Tese submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 29 de junho de 2018 por:

Profª. Drª. Lucia Catabriga (Orientador)
UFES/ES

Prof. Dr. Isaac Pinheiro dos Santos (Coorientador)
UFES/ES

Profª. Drª. Andrea Maria Pedrosa Valli (Examinador Externo)
UFES/ES

Profª. Drª. Maria Claudia Silva Boeres (Examinador Interno)
UFES/ES

Profª. Drª. Sandra Mara Cardoso Malta (Examinador Externo)
LNCC/RJ

Profª. Drª. Regina Célia Cerqueira de Almeida(Examinador Externo)
LNCC/RJ

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

Vitória-ES, 29 de junho de 2018.

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

Bento, Sérgio Souza, 1978-
B478n Nonlinear multiscale viscosity methods and time integration schemes for solving compressible Euler equations / Sérgio Souza Bento. – 2018.
93 f. : il.

Orientador: Lucia Catabriga.
Coorientador: Isaac Pinheiro dos Santos.
Tese (Doutorado em Ciência da Computação) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Método dos elementos finitos. 2. Rigidez. 3. Métodos estabilizados multiescala. 4. Euler, Equações de. 5. Funções bolha. I. Catabriga, Lucia. II. Santos, Isaac Pinheiro dos. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

Elaborada por Sandra Mara Borges Campos – CRB-6 ES-000593/O

To my mother Joana, and to my daughter Alana

Acknowledgements

First and foremost, I would like to thank God for giving me the strength and knowledge to carry out this work.

I express my gratitude to my family, specially my mom Joana, for their encouragement and support.

I would like to thank Isaac P. Santos by numerous discussions that contributed greatly to the elaboration of this work.

I would like to thank Lucia Catabriga for acting as supervisor for my thesis.

I would like to thank Leonardo Muniz de Lima, my big partnership in this academic journey.

I also thank my post-graduate colleagues: Paulo Wander, Ramoni, and Riedson.

Abstract

In this work we present nonlinear multiscale finite element methods for solving compressible Euler equations. The formulations are based on the strategy of separating scales – the core of the variational multiscale (finite element) methodology. The subgrid scale space is defined using bubble functions that vanish on the boundary of the elements, allowing to use a local Schur complement to define the resolved scale problem. The resulting numerical procedure allows the fine scales to depend on time. The formulations proposed in this work are residual based considering different ways for the artificial viscosity to act on all scales of the discretization. In the first formulation a nonlinear operator is added on all scales whereas in the second different nonlinear operators are included on macro and micro scales. We evaluate the efficiency of the formulations through numerical studies, comparing them with the SUPG combined with the shock-capturing operator YZβ and the CAU methodologies. Another contribution of this work concerns the time integration procedure. Density-based schemes suffer with undesirable effects of low speed flow including low convergence rate and loss of accuracy. Due to this phenomenon, local preconditioning is applied to the set of equations in the continuous case. Another alternative to solve this deficiency consists of using time integration methods with a stiff decay property. For this purpose, we propose a predictor-corrector method based on Backward Differentiation Formulas (BDF) that is not defined in the traditional sense found in the literature, i.e., using a predictor based on extrapolation.

Keywords: Finite element method, Multiscale formulation, Bubble function, Compressible flow problems, BDF methods, Stiff decay property, Euler equations.

Resumo

Este trabalho apresenta duas formulações do método de elementos finitos, utilizando estabilização multiescala, para resolver o sistema de equações de Euler compressíveis bidimensionais em variáveis conservativas. O espaço submalha é definido através de funções polinomiais que se anulam na fronteira dos elementos, conhecidas como funções bolha, permitindo o uso de um complemento de Schur local para definir o problema das escalas resolvidas. Esse procedimento resulta em uma metodologia numérica que permite variações temporais das escalas não resolvidas. As formulações propostas neste trabalho são baseadas em resíduo e consideram viscosidade artificial agindo em todas as escalas de discretização. Na primeira formulação um operador não linear é adicionado sobre todas as escalas, já na segunda formulação diferentes operadores não lineares são incluídos sobre as escalas macro e micro. A eficiência das novas formulações são avaliadas através de estudos numéricos, comparando-as com outras formulações, tais como os métodos SUPG combinado com o operador de captura de choque YZβ e CAU. Outra contribuição que este trabalho apresenta diz respeito ao avanço no tempo, uma vez que métodos baseados em densidade sofrem com efeitos indesejados em escoamento com baixa velocidade, o que inclui convergência lenta e perda de acurácia. Devido a esse fenômeno, a técnica de precondicionamento local é aplicada às equações no caso contínuo. Uma alternativa para resolver esta deficiência consiste em utilizar esquemas de avanço no tempo com propriedade de decaimento como L-estabilidade. Com esse intuito é proposto um esquema preditor-corretor baseado em *Backward Differentiation Formulas* (BDF) cuja predição é realizada através de extrapolação.

Palavras-chave: Elementos Finitos, Métodos Estabilizados Multiescala, Funções bolha, métodos BDF, Rigidez, Equações de Euler.

List of Figures

List of Tables

Contents

1 Introduction

A fluid is defined as a substance that continuously deforms under the action of a shear stress, no matter how small. Liquids and gases are called fluids because they can be made to flow, or move. Fluid mechanics is an important research area concerned with the mechanics of fluids and the forces on them. The study of movement of liquids and gases, using partial differential equations, describes how fluids behave and how they interact with their surrounding environment. The applicability of this subject is vast, including mechanical engineering, civil engineering, chemical engineering, biomedical engineering, geophysics, metereology, astrophysics, and biology. Fluid dynamics is one of the areas of fluid mechanics that study the effect of forces on fluid motion. Computational fluid dynamics (CFD) is a branch of fluid dynamics devoted to use numerical metodologies and data structures for solving fluid mechanics problems, by simulating physical phenomena such as ocean currents, predicting weather patterns, plate tectonics and even blood circulation. Some important technological applications of computational fluid dynamics include design of rocket engines, wind turbines, oil and gas pipelines (Fox et al., 2004; Graebel, 2007).

Flow can be classified as laminar or turbulent. Laminar flows are smoother whereas turbulent flows are characterized mainly by chaotic behavior. The viscosity and thickness of a fluid are important in determining the flow regime, where high viscosity implies laminar flow. The Reynolds number (Re) and the Mach number (M) are used to classify the flow regime. The Re is defined as the ratio of inertial to viscous forces. The inertial force is the fluid's resistance to change of motion, and the viscous force is the amount of friction due to the viscosity or thickness of the fluid. At low Re, the flow tends to be smooth, or laminar, while at high Re, the flow tends to be turbulent, forming eddies and vortices. The M characterizes the ratio of the velocity of a fluid to the velocity of sound in that fluid. In the case of an object moving through a fluid, such as an aircraft in flight, the Mach number is equal to the velocity of the object relative to the fluid divided by the velocity of sound in that fluid. Mach numbers less than one indicate subsonic flow; those greater than one, supersonic flow. Fluid flow, in addition, is classified as compressible, when $M > 0.3$, or incompressible, when $M < 0.3$ (Anderson, 2003; Fox et al., 2004; Graebel, 2007).

Numerical methods to solve compressible flow problems, modeled by the Euler equations, can be complicated by the presence of shocks and boundary layers in the computational domain. Since the beginning of the 1980s, researchers have been working on the development of numerical formulations based on stabilized finite element method to solve this difficulty (Brooks and Hughes, 1982; Hughes and Tezduyar, 1984; Rispoli et al., 2007; Tezduyar and Senga, 2007; John and Knobloch, 2007; Catabriga et al., 2009). Stabilization techniques are attempts to prevent numerical oscillations and other

instabilities when solving problems with high Reynolds and/or Mach numbers and shocks or strong boundary layers (Hughes, 1995; Franca et al., 1998; Tezduyar, 2004; Rispoli et al., 2007; Nassehi and Parvazinia, 2009; Gravemeier et al., 2010). In (Hughes et al., 2010), Hughes and co-workers provided a historical perspective on stabilized methods for computing compressible flow.

One well-known stabilized method for obtaining accurate solutions to the compressible Euler equations is the Streamline Upwind Petrov-Galerkin (SUPG) method coupled with the YZβ shock-capturing operator proposed by Tezduyar and Senga (2006). The stabilization parameter of the YZβ shock-capturing operator is calculated in a adaptive way, taking into account the directions of high gradients and the spatial discretization domain. The resulting stabilization parameter acts adaptively and is useful in avoiding excessive viscosity; this helps to maintain smaller numerical dissipation.

Hughes (1995) showed that stabilized methods could be obtained from the Variational Multiscale (VMS) framework. Variational multiscale finite element methods are based on the following idea: the discretization of the equations should be able to represent the behavior of all scales present in problems having a multiscale aspect, problems that are ubiquitous in science and engineering (Hughes, 1995; Hughes et al., 2004). The basic feature of this methodology is that the problem may be split into coarse and subgrid (or fine) scales sub-problems. The fine scale sub-problem is solved (or modeled) and used to modify the coarse scale equation such that the fine scale behavior is taken into account.

Guermond (2001) developed a multiscale method for convection-dominated transport problems in which a linear operator of artificial diffusion acting only on the small scales is added to the numerical formulation. However, like most stabilized methods, the method developed in (Guermond, 2001) requires a tunable parameter whose selection is a tricky task for actual problems. Following the idea proposed in (Guermond, 2001), Santos and co-workers (Santos and Almeida, 2007; Santos et al., 2012) presented the Nonlinear Subgrid Scale (NSGS) method in order to avoid user-defined coefficients. The definition of the nonlinear diffusion operator relies on the assumption that the velocity field may also be decomposed into fine and coarse scales, and the subgrid velocity field is then used to determine the amount of fine-scale artificial diffusion that is able to dissipate the kinetic energy at the smallest scales.

In the context of the variational multiscale stabilized formulation for advection diffusion equations, the idea of adding the same nonlinear diffusion in all scales of the discretization was considered in (Arruda et al., 2010; Valli et al., 2017). They proposed the Dynamic Diffusion (DD) method, in which the fine space can be constructed by bubble functions defined into elements; the amount of nonlinear diffusion is similar to the NSGS method. An extension of the DD method to the compressible Euler equations was presented in (Sedano et al., 2015). Although, the extended-DD method offered good

results, it failed to outperform either the Consistent Approximate Upwind Petrov-Galerkin (CAU) (Galeão and Carmo, 1988; Almeida and Galeão, 1996) method or the SUPG + YZβ method (Tezduyar and Senga, 2006).

In order to solve the compressible Euler equations, Bento et al. (2016) and Bento et al. (2017) modified the extended-DD method bringing forth two new methodologies, called Nonlinear Multiscale Viscosity methods, named NMV1 and NMV2. The NMV1 method adds a nonlinear dissipative operator in all scales, in which the amount of artificial viscosity is given by the YZβ shock-capturing viscosity parameter, since this method was designed specially to solve this kind of problem. In the NMV2 method, different nonlinear operators are included on micro and macro scales. The nonlinear dissipative operator acting on the unresolved (or fine) scale of the discretization is similar to that presented in (Santos and Almeida, 2007). The numerical model is completed by adding the YZβ shock-capturing operator modified on the resolved scale, taking into account the Mach number of the problem. It is worth pointing out that both formulation are self-adaptive and parameter-free, properties inherited from the NSGS, DD and YZβ methods. The NMV1 and NMV2 methods are compared with the CAU and SUPG + YZβ for the solution of problems from subsonic up to supersonic, providing good results.

Flows at a low speed demonstrate an incompressible behavior, because the density variation is almost negligible. Therefore, there are many challenges in developing numerical methods for solving problems from low to high speed compressible flows. Numerical methods addressed for solving low speed are usually pressure-based, since the flow is approaching to the incompressibility. On the other hand, in transonic and supersonic regimes the numerical methods generally are density-based. It is known that density-based strategy to solve compressible flow suffers severe deficiencies when applied to very low Mach number problems, degrading convergence speeds, and impacting the efficiency and accuracy of the numerical formulations (Li and Xiang, 2013). In the low Mach number limit the system of Euler equations becomes stiff due to large disparity in the timescales (Bassi et al., 2009).

With some adjustments, numerical methods can handle the full spectrum of speeds, as well as situations where the density does not change. For example, the work of Mittal and Tezduyar (1998) presents two formulations, one for compressible and another for incompressible flows, in the same algorithm, tuned by a parameter that depends on the Mach number. Wong et al. (2001) propose a specific construction of the stabilization matrix, using entropy variables, for a finite element SUPG formulation of the steady-state Euler equations. Local preconditioning or mass matrix preconditioning schemes have been proposed as a way to address this drawback using density-based method for low-Mach number flow, whose goal is to get an uniformization of the eigenvalues, smoothing the discrepancy of the time scales (Choi and Merkle, 1993; Lee, 1998; Colin et al., 2011;

Ginard et al., 2016a). Local preconditioning is applied to the set of continuous differential equations premultiplying the time derivative by a suitable preconditioning matrix. However, the original problem and the preconditioned one have different time evolution but the same steady-state solution. The application of these methodologies to unsteady problems requires the use of the "dual-time-stepping" technique (Lopez et al., 2012), in which the physical time derivative terms are treated as source and/or reactive terms.

According to Colin et al. (2011), local preconditioning approaches can be divided into three groups. The first one is based on the artificial compressibility method of Chorin (1965) which inspired the preconditioning method by Turkel (1987), for incompressible and low speed compressible flow. The Turkel method uses entropy as the dependent variable. The second group includes the works of Choi and Merkle (1993), Weiss and Smith (1995), Venkateswaran and Merkle (1999) and Briley et al. (2003). These methods are based on the temperature as the dependent variable. The Choi-Merkle (CM) preconditioner, presented for low Mach number, is suitable for Euler and Navier-Stokes equations by changing a single parameter and was extended to transient flows in (Nigro et al., 1998). The Weiss-Smith (WS) preconditioner was proposed to solve incompressible and compressible flows in transonic and low-speed regimes. Finally, as example of the third group is the Van Leer-Lee-Roe (VLR) preconditioner, proposed in (Leer et al., 1991) for Euler steady flow and extended to Navier-Stokes equations in (Lee, 1996). The VLR preconditioner is symmetric and optimal, in the sense that it equalizes the eigenvalues of the problem for all Mach number regimes.

Besides the reduction of the stiffness of the system of equations, local preconditioning also improves accuracy at low speed and the convergence of the numerical formulation. However, the major drawback of these methodologies is their reduced capacity to perform robust computations in stagnation point regions difficulting the use in an industrial context (Colin et al., 2011). To overcome these robustness issues, Colin et al. (2011) have studied a robust low speed preconditioning formulation for viscous flows, based on the WS and CM preconditioners, and called it WSCM preconditioner.

Most of the work inherent to local preconditioning is based on finite volume and finite difference methods. As far as we know, in the context of finite element, there are just a few works as described as follows. Nigro et al. (Nigro et al., 1997; Nigro et al., 1998) applied the CM preconditioner for solving steady compressible viscous flow; Lopez et al. (2012) extended the CM preconditioner to unsteady flow problems; Ginard et al. (Ginard et al., 2016a; Ginard et al., 2016b) applied the CM and VLR preconditioners for solving the Euler compressible steady flow.

Considering that density-based schemes suffer with undesirable effects of low speed flow, including low convergence speed and loss of accuracy, we present the NMV methods locally preconditioned for solving steady compressible Euler equations under low Mach

number regime. We apply two local preconditioning techniques to the NMV methods: VLR and WSCM.

An alternative way to deal with the difficulty of solving low speed flow is to use time integration methods with a decay stiff property. A class of methods based on backward differences, for stiff problems, was discovered by Curtiss and Hirschfelder (1952), and their importance for this kind of problems has been recognized in the work of Gear (1971). These methods belong to a class of implicit multistep time integrators known as the Backward Differentiation Formulas (BDF). We propose a predictor-corrector method based on BDF that is not defined in the traditional sense found in the literature, i.e., using a prediction based on extrapolation, as done in (Hay et al., 2015) for incompressible Navier-Stokes equations.

1.1 Contributions of this thesis

This thesis presents a set of important improvements in the solution of the Euler equations by the finite element method. We propose two numerical multiscale formulations to solve inviscid compressible flow problems in conservative variables, in which the fine space is constructed by bubble functions defined into the elements. The formulations are residual-based and consider artificial viscosity acting in all scales of the discretization. In the first formulation a nonlinear operator is added on all scales whereas in the second different nonlinear operators are included on macro and micro scales, taking into account the Mach number. It is worth pointing out that these formulations are self-adaptive and free of stabilization parameters, a property inherited from the YZβ and NSGS methods. Also, in order to solve steady inviscid compressible flow problems at low Mach number regime, we resort the local preconditioning approach, applying it in the proposed methods. Furthermore, we propose a predictor-corrector method based on second-order Backward Differentiation Formulas for our multiscale methodology.

1.2 Thesis outline

This thesis is organized as follows. In Chapter 2 we present a discussion of the governing equations and numerical formulations. In Chapter 3 we propose two multiscale formulations and reports the numerical experiments that are carried out to show how our formulations behave in a variety of transonic and supersonic flow problems. In Chapter 4 we propose a predictor-corrector scheme based on BDF-2 method and numerical results. In Chapter 5 we present two local preconditioners combined with multiscale approach. Finally, in Chapter 6 we present our conclusions and future works.

1.3 Publications related to this thesis

This section presents the publications related to the thesis research.

- R. Z. Sedano, **S. S. Bento**, L. M. Lima, L. Catabriga, *Predictor-Multicorrector Schemes for the Multiscale Dynamic Diffusion Method to Solve Compressible Flow Problems*. XXXVI Ibero-Latin American Congress on Computational Methods in Engineering (CILAMCE 2015), 2015;

- **S. S. Bento**, L. M. Lima, R. Z. Sedano, L. Catabriga, and I. P. Santos, *A nonlinear multiscale viscosity method to solve compressible flow problems*. 16th International Conference on Computational Science and Its Applications (ICCSA 2016), 4–7 July, 2016 - Beijing, China; **(ICCSA 2016 Best Paper Award)**

- **S. S. Bento**, R. Z. Sedano, L. M. Lima, L. Catabriga, and I. P. Santos, *Comparative Studies to Solve Compressible Problems by Multiscale Finite Element Methods*. Proceeding Series of the Brazilian Society of Computational and Applied Mathematics (CNMAC 2016), 2016;

- **S. S. Bento**, I. P. Santos, L. Catabriga, L. M. Lima, and A. Valli, *A Predictor-Multicorrector Scheme based on the BDF2 for Multiscale Finite Element Method for Compressible Flow Problems*. 19th International Conference on Finite Elements in Flow Problems (FEF 2017), 5–7 April, 2017 - Rome, Italy;

- **S. S. Bento**, P. W. Barbosa, I. P. Santos, L. M. Lima, and L. Catabriga, *A Nonlinear Finite Element Formulation Based on Multiscale Approach to Solve Compressible Euler Equations*. 17th International Conference on Computational Science and Its Applications (ICCSA 2017), 3–6 July, 2017 - Trieste, Italy;

- R. Baptista, **S. S. Bento**, I. P. Santos, L. M. Lima, A. Valli, and L. Catabriga, *A Multiscale Finite Element Formulation for the Incompressible Navier-Stokes Equations*. 18th International Conference on Computational Science and Its Applications (ICCSA 2018), 2–5 July, 2018 - Melbourne, Australia;

2 Governing equations and numerical formulations

In this chapter we present the Euler equations, the governing equations of inviscid compressible flow of a perfect gas. Additionally, the numerical formulations used for solving the system of Euler equations, such as the Galerkin finite element, the stabilized Streamline-Upwind/Petrov-Galerkin (SUPG), the SUPG method combined with YZβ shock-capturing, and the Consistent Approximate Upwind (CAU) methods are described here.

2.1 Euler equations in conservative variables

We consider the two-dimensional compressible Euler equations for an ideal gas. The equations may be written in conservative variables without source terms as a system of conservation laws,

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \mathbf{0}, \quad \text{in} \quad \Omega \times (0, t_f], \tag{2.1}$$

where t_f is a positive real number, representing the final time and Ω is a domain in $\mathbb{R}^2$, with boundary Γ, $\mathbf{U} \in \mathbb{R}^4$ is the vector of conservative variables, and $\mathbf{F}(\mathbf{U}) \in \mathbb{R}^{4\times 2}$, is the Euler flux vector (Noelle et al., 2014). Here,

$$\mathbf{U} = \underbrace{\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}}_{\text{conservative variables}} = \underbrace{\rho \begin{bmatrix} 1 \\ u \\ v \\ E \end{bmatrix}}_{\text{primitive variables}}, \tag{2.2}$$

where ρ is the fluid density, $\mathbf{u} = [u \quad v]^T$ is the velocity vector, ρE is the total energy, and E is the total specific energy. Others important physical quantities are the pressure p and the Mach number, $M = \dfrac{||\mathbf{u}||_2}{c}$, where $c = \sqrt{\gamma \dfrac{p}{\rho}}$ is the speed of sound, with $\gamma = \dfrac{c_p}{c_v}$ ($\gamma > 1$) being the ratio of specific heats, and c_p and c_v are the coefficients of specific heat at constant pressure and volume, respectively. The system of equations (2.1) is closed by the equation of state for pressure

$$p = (\gamma - 1)\left(\rho E - \frac{\rho}{2}||\mathbf{u}||_2^2\right). \tag{2.3}$$

Alternatively, Eq. (2.1) can be rewritten as in the quasi-linear form:

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A_x}\frac{\partial \mathbf{U}}{\partial x} + \mathbf{A_y}\frac{\partial \mathbf{U}}{\partial y} = \mathbf{0}, \quad \text{in} \quad \Omega \times (0, t_f], \tag{2.4}$$

where $\mathbf{A}_x = \dfrac{\partial \mathbf{F}_x}{\partial \mathbf{U}}$ and $\mathbf{A}_y = \dfrac{\partial \mathbf{F}_y}{\partial \mathbf{U}}$ are the Jacobian matrices. Associated with Eq.(2.4) we have an appropriate set of boundary and initial conditions. We assume the following boundary and initial conditions,

$$\mathbf{B}\mathbf{U} = \mathbf{Z}, \quad \text{on } \Gamma \times (0, t_f], \tag{2.5}$$

$$\mathbf{U}(\mathbf{x}, t) = \mathbf{U}_0, \tag{2.6}$$

where $\mathbf{B}$ denotes a general boundary operator, and $\mathbf{Z}$ and $\mathbf{U}_0$ are given functions.

2.2 Jacobian matrices of the Euler flux

The Jacobian matrices of the Euler fluxes, $\mathbf{A}_x$ and $\mathbf{A}_y$, are described as follow

$$\mathbf{A}_x = \begin{bmatrix} \dfrac{\partial \mathbf{F}_x}{\partial U_1} & \dfrac{\partial \mathbf{F}_x}{\partial U_2} & \dfrac{\partial \mathbf{F}_x}{\partial U_3} & \dfrac{\partial \mathbf{F}_x}{\partial U_4} \end{bmatrix} \quad \text{and} \quad \mathbf{A}_y = \begin{bmatrix} \dfrac{\partial \mathbf{F}_y}{\partial U_1} & \dfrac{\partial \mathbf{F}_y}{\partial U_2} & \dfrac{\partial \mathbf{F}_y}{\partial U_3} & \dfrac{\partial \mathbf{F}_y}{\partial U_4} \end{bmatrix}.$$

The Euler flux vector can be written as

$$\mathbf{F}(\mathbf{U}) = \underbrace{\begin{bmatrix} \mathbf{U}_{23} \\ \frac{1}{U_1}\mathbf{U}_{23} \otimes \mathbf{U}_{23} + p\mathbf{I} \\ (U_4 + p)\dfrac{\mathbf{U}_{23}}{U_1} \end{bmatrix}}_{\text{conservative variables}} = \underbrace{\begin{bmatrix} \rho\mathbf{u} \\ \rho\mathbf{u} \otimes \mathbf{u} + p\mathbf{I} \\ (\rho E + p)\mathbf{u} \end{bmatrix}}_{\text{primitive variables}} \tag{2.7}$$

where $\mathbf{U}_{23} = [U_2 \;\; U_3]^T$, the operator $\otimes$ is the tensor product in $\mathbb{R}^2$ and the pressure can be calculated in the conservative variables as

$$p = (\gamma - 1)\left(U_4 - \frac{||\mathbf{U}_{23}||_2^2}{2U_1}\right).$$

The Jacobian matrix $\mathbf{A}_x$ in conservative variables is given by

$$\mathbf{A}_x = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\left(\dfrac{U_2}{U_1}\right)^2 + (\gamma-1)\dfrac{||\mathbf{U}_{23}||_2^2}{2U_1^2} & (3-\gamma)\dfrac{U_2}{U_1} & -(\gamma-1)\dfrac{U_3}{U_1} & (\gamma-1) \\ -\dfrac{U_2U_3}{U_1^2} & \dfrac{U_3}{U_1} & \dfrac{U_2}{U_1} & 0 \\ -\dfrac{U_2}{U_1^2}\left[\gamma U_4 - (\gamma-1)\dfrac{||\mathbf{U}_{23}||_2^2}{U_1}\right] & \gamma\dfrac{U_4}{U_1} - \dfrac{(\gamma-1)}{2U_1^2}\left(||\mathbf{U}_{23}||_2^2 + 2U_2^2\right) & -(\gamma-1)\dfrac{U_2U_3}{U_1^2} & \gamma\dfrac{U_2}{U_1} \end{bmatrix} \tag{2.8}$$

and the Jacobian matrix $\mathbf{A}_y$ in conservative variables is given by

$$\mathbf{A}_y = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -\dfrac{U_2U_3}{U_1^2} & \dfrac{U_3}{U_1} & \dfrac{U_2}{U_1} & 0 \\ -\left(\dfrac{U_3}{U_1}\right)^2 + (\gamma-1)\dfrac{||\mathbf{U}_{23}||_2^2}{2U_1^2} & -(\gamma-1)\dfrac{U_2}{U_1} & (\gamma-1)\dfrac{U_3}{U_1} & (\gamma-1) \\ -\dfrac{U_3}{U_1^2}\left[\gamma U_4 - (\gamma-1)\dfrac{||\mathbf{U}_{23}||_2^2}{U_1}\right] & -(\gamma-1)\dfrac{U_2U_3}{U_1^2} & \gamma\dfrac{U_4}{U_1} - \dfrac{(\gamma-1)}{2U_1^2}(||\mathbf{U}_{23}||_2^2 + 2U_3^2) & \gamma\dfrac{U_3}{U_1} \end{bmatrix}. \tag{2.9}$$

2.3 Numerical formulations

The numerical formulations presented here to solve the problem (2.1) are based on the finite element method in space and finite differences in time. The finite element method is not applied directly in the classical form of the problem. For this, a variational (or weak) formulation of the model must be defined. The weak form of the mathematical model relaxes the regularity requirements of the solution. It is obtained by multiplying the differential equation (2.1) by a test function which vanishes on the part of the boundary prescribed with Dirichlet boundary condition and integrating over the entire domain Ω, to get an integral formulation. For the problem (2.1), the variational formulation reads: for each $t \in (0, t_f]$ such that $\mathbf{U}(\mathbf{x}, 0) = \mathbf{U}(\mathbf{x})$, find $\mathbf{U} \in \mathcal{V}_Z$ such that

$$\int_\Omega \mathbf{W} \cdot \left(\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}_x \frac{\partial \mathbf{U}}{\partial x} + \mathbf{A}_y \frac{\partial \mathbf{U}}{\partial y} \right) \, d\Omega = \mathbf{0}, \forall \ \mathbf{W} \in \mathcal{V}_0, \tag{2.10}$$

where $\mathcal{V}_Z$ is the vectorial solution function space defined as

$$\mathcal{V}_Z = \{\mathbf{U} \in [H^1(\Omega)]^4; \quad \mathbf{B}\mathbf{U}_h = \mathbf{Z} \ \text{ on } \ \Gamma_D\},$$

with Γ_D representing the part of the boundary Γ prescribed with Dirichlet boundary conditions. The weighting function space $\mathcal{V}_0$ is equal to $\mathcal{V}_Z$ with $\mathbf{Z} = \mathbf{0}$ on Γ_D.

To define the finite element method, we consider a triangular partition $\mathcal{T}_h$ of the domain Ω into *nel* elements, where

$$\Omega = \bigcup_{e=1}^{nel} \Omega_e \quad \text{with} \quad \Omega_i \cap \Omega_j = \varnothing, \quad \text{for} \quad i, j = 1, 2, \ldots, nel \quad \text{and} \quad i \neq j.$$

The infinite dimensional space $\mathcal{V}_Z$ is replaced by the finite dimensional subspace

$$\mathcal{V}_{Zh} = \{\mathbf{U}_h \in [H^1(\Omega)]^4; \quad \mathbf{U}_h|_{\Omega_e} \in [\mathbb{P}_1(\Omega_e)]^4, \mathbf{B}\mathbf{U}_h = \mathbf{Z} \ \text{ on } \ \Gamma_D\}, \tag{2.11}$$

with $\mathbb{P}_1(\Omega_e)$ representing the set of first order polynomials in Ω_e, and $H^1(\Omega)$ denotes the Sobolev space of square-integrable functions whose first derivatives are also square-integrable (Brenner and Scott, 2002). The finite dimensional space (2.11) is called finite element space.

For each $t \in (0, t_f]$ such that $\mathbf{U}(\mathbf{x}, 0) = \mathbf{U}(\mathbf{x})$, the Galerkin Finite Element formulation for solving Euler equation, Eq. (2.4), consists of finding $\mathbf{U}_h \in \mathcal{V}_{Zh}$ such that

$$\int_\Omega \mathbf{W}_h \cdot \left(\frac{\partial \mathbf{U}_h}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_h}{\partial y} \right) \, d\Omega = \mathbf{0}, \forall \ \mathbf{W}_h \in \mathcal{V}_{0h}. \tag{2.12}$$

It is well known that the standard Galerkin finite element method is not suitable for convection-dominated problems that presents internal and/or boundary layers (Brooks and Hughes, 1982). Therefore, it is necessary to use some stabilization method in order to avoid spurious oscillations in the solution. In the next sections we present a class of stabilized finite element formulations for solving convection-dominated problems.

2.3.1 Streamline Upwind Petrov-Galerkin (SUPG) method

One of the first successful and most popular stabilized finite element method for flow and transport problems is the Streamline-Upwind/Petrov-Galerkin (SUPG) formulation. It was introduced for advection–diffusion problems and incompressible flow in (Brooks and Hughes, 1982) and for compressible flows, in the context of conservative variable, in [REFERENCE (Tezduyar and Hughes 1982, Tezduyar and Hughes 1983)]. The SUPG method introduces, on the element-level, an artificial diffusion only in the streamlines direction. The method consists of finding $\mathbf{U}_h(\mathbf{x},t) \in \mathcal{V}_{Zh}$ for $t \in (0,t_f]$ such that $\mathbf{U}(\mathbf{x},0) = \mathbf{U}(\mathbf{x})$ and

$$\int_\Omega \mathbf{W}_h \cdot \left(\frac{\partial \mathbf{U}_h}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_h}{\partial y} \right) \, d\Omega$$
$$+ \sum_{e=1}^{nel} \int_{\Omega_e} \boldsymbol{\tau}_{supg} \left(\mathbf{A}_x^h \frac{\partial \mathbf{W}_h}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{W}_h}{\partial y} \right) \cdot R(\mathbf{U}_h) \, d\Omega = \mathbf{0}, \quad \forall \ \mathbf{W}_h \in \mathcal{V}_{0h}, \tag{2.13}$$

where

$$R(\mathbf{U}_h) = \frac{\partial \mathbf{U}_h}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_h}{\partial y} \tag{2.14}$$

is the residue of the Euler equations, evaluated on each element Ω_e and

$$\boldsymbol{\tau}_{supg} = \tau \mathbf{I} \tag{2.15}$$

is the stabilization parameter, with $\mathbf{I}$ denoting the 4×4 identity tensor and

$$\tau = max[0, \tau_t + \zeta(\tau_a - \tau_\delta)]. \tag{2.16}$$

The parameter ζ in (2.16) is given by

$$\zeta = \frac{2\alpha CFL}{1 + 2\alpha CFL} \tag{2.17}$$

and the stabilization parameters τ_t and τ_a are defined as

$$\tau_t = \frac{2}{3(1 + 2\alpha CFL)} \tau_a$$

and

$$\tau_a = \frac{h}{2(c + |\mathbf{u} \cdot \beta|)},$$

corresponding to the time-dependent and convective terms, respectively. In addition, the parameter

$$\tau_\delta = \frac{\delta_{shock}}{(c + |\mathbf{u} \cdot \beta|)^2}$$

is used to remove excessive effects of the shock-capturing operator, with δ_{shock} denoting a shock-capturing parameter (it will be set later). In Eq. (2.17), α is a time integration parameter and CFL is the *Courant-Friedrichs-Lewy* number given by

$$CFL = \frac{(c + |\mathbf{u} \cdot \beta|)\Delta t}{h},$$

where Δt is the time step and the unit vector β is defined as

$$\beta = \frac{\nabla||\mathbf{U}||_2^2}{||\nabla||\mathbf{U}||_2^2||_2},$$

with $h = \sqrt{2A^e}$ denoting the element length and A^e the element area.

2.3.2 The SUPG method with YZβ shock-capturing - SUPG+YZβ

The SUPG method presents properties of good stability and accuracy, if the exact solution is smooth. However, for problems whose solution is not smooth, spurious oscillations can remain in subregions with sharp internal and boundary layers. Therefore, shock-capturing methods have been used together with the SUPG method, in order to control the solution gradient in other directions, preventing oscillations in regions with boundary layers. The SUPG formulation combined with YZβ shock-capturing operator (Tezduyar and Senga, 2006) for Euler equation consists of finding $\mathbf{U}_h(\mathbf{x},t) \in \mathcal{V}_{Zh}$ for $t \in (0,t_f]$ such that

$$\begin{aligned}
&\int_\Omega \mathbf{W}_h \cdot \left(\frac{\partial \mathbf{U}_h}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_h}{\partial y} \right) d\Omega \\
+ &\sum_{e=1}^{nel} \int_{\Omega_e} \boldsymbol{\tau}_{supg} \left(\mathbf{A}_x^h \frac{\partial \mathbf{W}_h}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{W}_h}{\partial y} \right) \cdot \left(\frac{\partial \mathbf{U}_h}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_h}{\partial y} \right) d\Omega \\
+ &\sum_{e=1}^{nel} \int_{\Omega_e} \delta_{yz\beta} \left(\frac{\partial \mathbf{W}_h}{\partial x} \cdot \frac{\partial \mathbf{U}_h}{\partial x} + \frac{\partial \mathbf{W}_h}{\partial y} \cdot \frac{\partial \mathbf{U}_h}{\partial y} \right) d\Omega = \mathbf{0}, \forall\ \mathbf{W}_h \in \mathcal{V}_{0h},
\end{aligned} \tag{2.18}$$

with initial condition $\mathbf{U}_h(\mathbf{x},0) = \mathbf{U}_0(\mathbf{x})$. The shock-capturing parameter, denoted by $\delta_{yz\beta}$, depends on the solution and is defined as (Tezduyar and Senga, 2006)

$$\delta_{yz\beta}(\mathbf{U}_h) = \|\mathbf{Y}^{-1} R(\mathbf{U}_h)\|_2 \left(\sum_{i=1}^{2} \left\| \mathbf{Y}^{-1} \frac{\partial \mathbf{U}_h}{\partial x_i} \right\|_2^2 \right)^{\frac{\beta}{2}-1} \|\mathbf{Y}^{-1}\mathbf{U}_h\|_2^{1-\beta} h_{yz\beta}^{\beta}, \tag{2.19}$$

where $R(\mathbf{U}_h)$, defined in Eq. (2.14), is the residue of the problem on Ω_e, $\mathbf{Y}$ is a diagonal matrix constructed from the reference values of the components of $\mathbf{U}$, given by

$$\mathbf{Y} = \begin{bmatrix} (U_1)_{\text{ref}} & & & \\ & (U_2)_{\text{ref}} & & \\ & & (U_3)_{\text{ref}} & \\ & & & (U_4)_{\text{ref}} \end{bmatrix}. \tag{2.20}$$

Catabriga et al. (2009) showed that using the reference values considered in the inflow presented better results. In our experiments we will use fixed reference values, those corresponding to the inflow boundaries. The local length scale, $h_{yz\beta}$, is defined as in (Tezduyar, 2004) by

$$h_{yz\beta} = \left(\sum_{a=1}^{3} |\mathbf{j} \cdot \nabla N_a| \right)^{-1}, \tag{2.21}$$

$\mathbf{j}$ is a unit vector defined as

$$\mathbf{j} = \frac{\nabla\rho}{\|\nabla\rho\|_2}$$

and N_a is the interpolation function associated with node a. It is important to note that, the local length $h_{yz\beta}$ is defined automatically taking into account the directions of high gradients and spatial discretization domain.

The formulation described in Eq. (2.18) results in a system of differential algebraic equations,

$$\boldsymbol{M}\dot{\mathbf{U}}_h + \boldsymbol{K}\mathbf{U}_h = \mathbf{0}, \tag{2.22}$$

where $\boldsymbol{U}_h$ is the vector of unknowns, whereas $\dot{\boldsymbol{U}}_h$ is the vector of time derivatives. The matrices $\boldsymbol{M}$ and $\boldsymbol{K}$ are defined as follow,

$$\boldsymbol{M} = \mathop{\mathbf{A}}_{e=1}^{nel}(\boldsymbol{M}^g_{hh} + \boldsymbol{M}^{supg}) \tag{2.23}$$

and

$$\boldsymbol{K} = \mathop{\mathbf{A}}_{e=1}^{nel}(\boldsymbol{K}^g_{hh} + \boldsymbol{K}^{supg} + \boldsymbol{K}^{yz\beta}), \tag{2.24}$$

with

$$\begin{aligned}
\boldsymbol{M}^g_{hh} &: \int_{\Omega_e} \mathbf{W}_h \cdot \frac{\partial \mathbf{U}_h}{\partial t}\, d\Omega;\\
\boldsymbol{M}^{supg} &: \int_{\Omega_e} \boldsymbol{\tau}_{supg}\left(\mathbf{A}^h_x\frac{\partial \mathbf{W}_h}{\partial x} + \mathbf{A}^h_y\frac{\partial \mathbf{W}_h}{\partial y}\right)\cdot\frac{\partial \mathbf{U}_h}{\partial t}\, d\Omega;\\
\boldsymbol{K}^g_{hh} &: \int_{\Omega_e} \mathbf{W}_h \cdot \left(\mathbf{A}^h_x\frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{A}^h_y\frac{\partial \mathbf{U}_h}{\partial y}\right)\, d\Omega;\\
\boldsymbol{K}^{supg} &: \int_{\Omega_e} \boldsymbol{\tau}_{supg}\left(\mathbf{A}^h_x\frac{\partial \mathbf{W}_h}{\partial x} + \mathbf{A}^h_y\frac{\partial \mathbf{W}_h}{\partial y}\right)\cdot\left(\mathbf{A}^h_x\frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{A}^h_y\frac{\partial \mathbf{U}_h}{\partial y}\right)\, d\Omega;\\
\boldsymbol{K}^{yz\beta} &: \int_{\Omega_e} \delta_{yz\beta}\left(\frac{\partial \mathbf{W}_h}{\partial x}\cdot\frac{\partial \mathbf{U}_h}{\partial x} + \frac{\partial \mathbf{W}_h}{\partial y}\cdot\frac{\partial \mathbf{U}_h}{\partial y}\right)\, d\Omega.
\end{aligned}$$

2.3.3 The Consistent Approximate Upwind (CAU) method

The CAU (Consistent Approximate Upwind) is a stabilized finite element method that introduces in a consistent way, besides the SUPG contribution, a discontinuity capturing term that controls the derivatives in the direction of the approximate solution (Galeão and Carmo, 1988; Almeida and Galeão, 1996). The CAU method for the Euler equation consists of finding $\mathbf{U}_h(\mathbf{x},t) \in \mathcal{V}_{Zh}$ for $t \in (0,t_f]$ such that

$$\begin{aligned}
&\int_\Omega \mathbf{W}_h \cdot \left(\frac{\partial \mathbf{U}_h}{\partial t} + \mathbf{A}^h_x\frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{A}^h_y\frac{\partial \mathbf{U}_h}{\partial y}\right)\, d\Omega\\
+ &\sum_{e=1}^{nel}\int_{\Omega_e} \boldsymbol{\tau}_{cau}\left(\mathbf{A}^h_x\frac{\partial \mathbf{W}_h}{\partial x} + \mathbf{A}^h_y\frac{\partial \mathbf{W}_h}{\partial y}\right)\cdot\left(\frac{\partial \mathbf{U}_h}{\partial t} + \mathbf{A}^h_x\frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{A}^h_y\frac{\partial \mathbf{U}_h}{\partial y}\right)\, d\Omega\\
+ &\sum_{e=1}^{nel}\int_{\Omega_e} \delta_{cau}\left(\frac{\partial \mathbf{W}_h}{\partial x}\cdot\frac{\partial \mathbf{U}_h}{\partial x} + \frac{\partial \mathbf{W}_h}{\partial y}\cdot\frac{\partial \mathbf{U}_h}{\partial y}\right)\, d\Omega = \mathbf{0}, \forall\ \mathbf{W}_h \in \mathcal{V}_{0h},
\end{aligned} \tag{2.25}$$

with initial condition $\mathbf{U}_h(\mathbf{x},0) = \mathbf{U}_0(\mathbf{x})$. The stabilization parameter $\boldsymbol{\tau}_{cau}$ is defined as the SUPG parameter, i.e.,

$$\boldsymbol{\tau}_{cau} = \boldsymbol{\tau}_{supg},$$

given in Eq. (2.15). The shock-capturing parameter is defined by

$$\delta_{cau} = \begin{cases} \dfrac{||R(\mathbf{U}_h)||_{\tilde{\mathbf{A}}_0^{-1}}}{||\nabla_\xi \mathbf{U}_h||_{\tilde{\mathbf{A}}_0^{-1}}}, & \text{if } ||\nabla \mathbf{U}_h||_{\tilde{\mathbf{A}}_0^{-1}} > tol_\delta, \\ 0, & \text{otherwise,} \end{cases} \tag{2.26}$$

where $R(\mathbf{U}^h)$ is the residue of the problem on Ω_e (Eq. (2.14)) and tol_δ is a small fixed real number. The norm $||\cdot||_{\tilde{\mathbf{A}}_0^{-1}}$ is defined as follow,

$$||\mathbf{W}||_{\tilde{\mathbf{A}}_0^{-1}} = \left(\mathbf{W}^T \tilde{\mathbf{A}}_0^{-1} \mathbf{W}\right)^{\frac{1}{2}},$$

where $\mathbf{W} \in \mathbb{R}^4$, and

$$||\nabla_\xi \mathbf{U}_h||_{\tilde{\mathbf{A}}_0^{-1}} = \left|\left|\frac{\partial x}{\partial \xi}\frac{\partial \mathbf{U}_h}{\partial x} + \frac{\partial y}{\partial \xi}\frac{\partial \mathbf{U}_h}{\partial y}\right|\right|_{\tilde{\mathbf{A}}_0^{-1}} + \left|\left|\frac{\partial x}{\partial \eta}\frac{\partial \mathbf{U}_h}{\partial x} + \frac{\partial y}{\partial \eta}\frac{\partial \mathbf{U}_h}{\partial y}\right|\right|_{\tilde{\mathbf{A}}_0^{-1}}. \tag{2.27}$$

The matrix $\widetilde{\mathbf{A}}_0^{-1}$ is the Jacobian of the transformation between the entropy and conservation variables (Shakib et al., 1991; Catabriga and Coutinho, 2002), given by

$$\tilde{\mathbf{A}}_0^{-1} = \frac{-1}{\rho i V_4}\begin{bmatrix} k_1^2+\gamma & k_1 V_2 & k_1 V_3 & (k_1+1)V_4 \\ & V_2^2 - V_4 & V_2 V_3 & V_2 V_4 \\ & & V_3^2 - V_4 & V_3 V_4 \\ sim & & & V_4^2 \end{bmatrix}, \tag{2.28}$$

where $k_1 = \dfrac{V_2^2 + V_3^2}{2V_4}$ and V_j, $j = 1,\ldots,4$ are entropy variables. Mapping $\mathbf{V} \mapsto \mathbf{U}$, we obtain

$$\mathbf{V} = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \frac{1}{\rho i}\begin{bmatrix} -U_4 + \rho i(\gamma + 1 - s) \\ U_2 \\ U_3 \\ -U_1 \end{bmatrix},$$

with $s = ln\left(\dfrac{(\gamma-1)\rho i}{U_1^\gamma}\right)$ and $\rho i = U_4 - \dfrac{||U_{23}||_2^2}{2U_1}$.

The formulation described in Eq. (2.25) results in a system of differential algebraic equations,

$$\boldsymbol{M}\dot{\mathbf{U}}_h + \boldsymbol{K}\mathbf{U}_h = \mathbf{0}, \tag{2.29}$$

where the matrices $\boldsymbol{M}$ and $\boldsymbol{K}$ are defined as in Eqs. (2.23) and (2.24).

3 Nonlinear multiscale viscosity methods

In this chapter we present a class of residual based nonlinear multiscale viscosity methods to solve compressible flows. In particular, we propose two numerical multiscale formulations in conservative variables, where the fine space is constructed by bubble functions defined into the elements. In the first formulation a nonlinear operator is added on all scales whereas in the second one different nonlinear operators are included on macro and micro scales, taking into account the Mach number.

3.1 Multiscale finite element discretization

The main idea of multiscale finite element methods consists in the enrichment of the solution space in order to take into account the small-scales phenomenology. Once, multiscale methodology has been used to build stable approximations, some researchers (Hughes, 1995; Franca and Farhat, 1995; Guermond, 1999) have considered the practical possibility of stabilizing convection dominated equations by means of bubble functions. In this work, we use bubble functions to build the small scale space.

In order to define the multiscale finite element methods, we introduce the function space $\mathcal{V}_{Zhb}$, which is written as the direct sum,

$$\mathcal{V}_{Zhb} = \mathcal{V}_{Zh} \oplus \mathcal{V}_b, \tag{3.1}$$

where the subspace $\mathcal{V}_{Zh}$ is defined in Eq. (2.11) and the subspace $\mathcal{V}_b$ is given by

$$\mathcal{V}_b = \{\mathbf{U}_b \in [H_0^1(\Omega)]^4 \quad | \quad \mathbf{U}_b|_{\Omega_e} \in [span(\psi_b)]^4, \quad \forall\ \Omega_e \in \mathcal{T}_h\}, \tag{3.2}$$

where $H_0^1(\Omega)$ is a space of function in $H^1(\Omega)$ that vanish at the boundary of Ω (Brenner and Scott, 2002) and ψ_b is a bubble function. For a given Ω_e, the bubble basis function ψ_b satisfies

$$\begin{aligned}
\psi_b(\mathbf{x}) &> 0, \forall \mathbf{x} \in \Omega_e; \\
\psi_b(\mathbf{x}) &= 0, \forall \mathbf{x} \in \partial\Omega_e; \\
\psi_b(\mathbf{x}) &= 1, \text{ at the barycenter of the triangle } \Omega_e.
\end{aligned}$$

Here, the bubble function is a cubic polynomial defined as

$$\psi_b(\mathbf{x}) = 27 N_1^e(\mathbf{x}) N_2^e(\mathbf{x}) N_3^e(\mathbf{x}), \tag{3.3}$$

where N_i^e represents the local shape function associated with node $i = 1, 2, 3$. The space $\mathcal{V}_h$ represents the resolved (coarse) scale space whereas $\mathcal{V}_b$ stands for the subgrid (fine) scale space (Fig. 3.1). The space defined in (3.1) with $\mathbf{Z} = \mathbf{0}$ on Γ_D is written as $\mathcal{V}_{0hb} = \mathcal{V}_{0h} \oplus \mathcal{V}_b$.

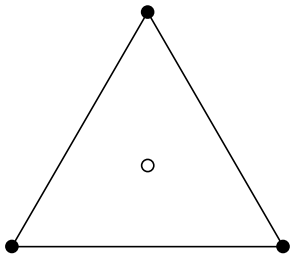


Figure 3.1 – $\mathcal{V}_{Zhb}$ Representation: • stands for $\mathcal{V}_{Zh}$ nodes and ○ stands for $\mathcal{V}_b$ nodes.

The nonlinear multiscale stabilization method for the Euler equations consists of finding $\mathbf{U}_{hb} = \mathbf{U}_h + \mathbf{U}_b \in \mathcal{V}_{Zhb}$, with $\mathbf{U}_h \in \mathcal{V}_{Zh},\ \mathbf{U}_b \in \mathcal{V}_b$, such that $\mathbf{U}_{hb}(\cdot, 0) = \mathbf{U}_0$, and

$$\begin{aligned} \int_\Omega \mathbf{W}_{hb} \cdot \left(\frac{\partial \mathbf{U}_{hb}}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_{hb}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_{hb}}{\partial y} \right) d\Omega \quad + \\ \sum_{e=1}^{nel} \int_{\Omega_e} \delta_\nu(\mathbf{U}_h) \left(\frac{\partial \mathbf{W}_{hb}}{\partial x} \cdot \frac{\partial \mathbf{U}_{hb}}{\partial x} + \frac{\partial \mathbf{W}_{hb}}{\partial y} \cdot \frac{\partial \mathbf{U}_{hb}}{\partial y} \right) d\Omega \quad = \quad \mathbf{0}, \quad \forall\ \mathbf{W}_{hb} \in \mathcal{V}_{0hb}, \end{aligned} \tag{3.4}$$

where $\mathbf{W}_{hb} = \mathbf{W}_h + \mathbf{W}_b \in \mathcal{V}_{0hb}$ with $\mathbf{W}_h \in \mathcal{V}_{0h}$, $\mathbf{W}_b \in \mathcal{V}_b$, and $\delta_\nu(\mathbf{U}_h)$ is a parameter that controls the amount of artificial viscosity and consequently defines the method. In general, as we can see in (Santos and Almeida, 2007; Arruda et al., 2010; Ginard et al., 2016b; Valli et al., 2017), this term is proportional to the residual of the Eq. (2.4) for the resolved solution, which means that the numerical dissipation is more effective in regions of the domain where the residual is significant.

3.2 The Dynamic Diffusion method

In the context of the variational multiscale stabilized formulation for advection diffusion equations, the idea of adding the same nonlinear diffusion in all scales of the discretization was considered in (Arruda et al., 2010; Valli et al., 2017). They proposed the nonlinear multiscale Dynamic Diffusion (DD) method that introduces the same amount of artificial diffusion onto both the resolved and unresolved scales, unlike the NSGS method (Santos and Almeida, 2007; Santos et al., 2012) in which a nonlinear operator of artificial diffusion is introduced only onto the subgrid scales. In (Werner et al., 2010; Valli et al., 2014; Valli et al., 2015) the DD method was applied for solving transient transport equations, and in (Mattos, 2012; Sedano et al., 2015) an extension of the DD method to solve compressible Euler equations was presented.

The DD method addressed for the Euler equations (Sedano et al., 2015) consists of

finding $\mathbf{U}_{hb} = \mathbf{U}_h + \mathbf{U}_b \in \mathcal{V}_{Zhb}$ with $\mathbf{U}_h \in \mathcal{V}_{Zh}$, $\mathbf{U}_b \in \mathcal{V}_b$ such that

$$\int_\Omega \mathbf{W}_{hb} \cdot \Big(\frac{\partial \mathbf{U}_{hb}}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_{hb}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_{hb}}{\partial y} \Big) d\Omega \quad +$$

$$\sum_{e=1}^{nel} \int_{\Omega_e} \delta_h^{DD}(\mathbf{U}_h) \Big(\frac{\partial \mathbf{W}_{hb}}{\partial x} \cdot \frac{\partial \mathbf{U}_{hb}}{\partial x} + \frac{\partial \mathbf{W}_{hb}}{\partial y} \cdot \frac{\partial \mathbf{U}_{hb}}{\partial y} \Big) d\Omega \quad = \quad \mathbf{0}, \quad \forall\ \mathbf{W}_{hb} \in \mathcal{V}_{0hb}, \qquad (3.5)$$

where $\mathbf{W}_{hb} = \mathbf{W}_h + \mathbf{W}_b \in \mathcal{V}_{0hb}$ with $\mathbf{W}_h \in \mathcal{V}_{0h}$, $\mathbf{W}_b \in \mathcal{V}_b$. The amount of artificial viscosity is calculated on the element-level through the function

$$\delta_h^{\mathrm{DD}}(\mathbf{U}_h) = \frac{1}{2} \mu(\hbar) \delta_{cau}, \qquad (3.6)$$

where δ_{cau} is defined in (2.26), $\mu(\hbar) = \sqrt{2A_e}$ is the element length, A_e is the element area.

The DD method offered good results for solving transport equations. However, for problem with shock the extended-DD method did not live up to expectations compared to the SUPG formulations with shock capturing operators as CAU and YZβ, as reported in (Sedano et al., 2015). The amount of artificial viscosity inserted by the extended-DD method is based on the parameter of viscosity of the CAU method. On the other hand, the operator of YZβ shock-capturing was designed specially for supersonic compressible flow, making it one of the best methods for this type of problem. This motivates using a similar approach in defining the amount of artificial diffusion in the context of multiscale methodology. Based on that, Bento et al. (2016) and Bento et al. (2017) modified the extended-DD method bringing forth two new methodologies, called Nonlinear Multiscale Viscosity methods, abbreviated by NMV1 and NMV2. These new methods will be presented in the next two sections.

3.3 The NMV1 method

Following the same philosophy of the DD method, adding artificial viscosity isotropically in all scales of the discretization, we propose a new formulation where the parameter that defines the amount of artificial viscosity is given by the YZβ shock-capturing viscosity parameter, as described in (Tezduyar and Senga, 2006). Indeed, since the YZβ method was designed specially to solve compressible flow, its stabilization parameter is enriched locally by a specific length scale that is calculated automatically, accounting for the direction of the gradient density solution. The combination of the YZβ parameter with the DD operator leads to a multiscale method with good properties of stability to solve inviscid flows. Also, the NMV1 method is self-adaptive and parameter-free, properties inherited from the DD and YZβ methods.

The NMV1 method for the Euler equation consists of finding $\mathbf{U}_{hb} = \mathbf{U}_h + \mathbf{U}_b \in \mathcal{V}_{Zhb}$

with $\mathbf{U}_h \in \mathcal{V}_{Zh}$, $\mathbf{U}_b \in \mathcal{V}_b$ such that

$$\underbrace{\int_\Omega \mathbf{W}_{hb} \cdot \Big(\frac{\partial \mathbf{U}_{hb}}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_{hb}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_{hb}}{\partial y}\Big) d\Omega}_{\text{Galerkin term}} \quad +$$

$$\underbrace{\sum_{e=1}^{nel} \int_{\Omega_e} \delta_h(\mathbf{U}_h) \Big(\frac{\partial \mathbf{W}_{hb}}{\partial x} \cdot \frac{\partial \mathbf{U}_{hb}}{\partial x} + \frac{\partial \mathbf{W}_{hb}}{\partial y} \cdot \frac{\partial \mathbf{U}_{hb}}{\partial y}\Big) d\Omega}_{\text{Nonlinear stabilization term in all scales}} = \mathbf{0}, \quad \forall\ \mathbf{W}_{0hb} \in \mathcal{V}_{hb}, \tag{3.7}$$

where $\mathbf{W}_{hb} = \mathbf{W}_h + \mathbf{W}_b \in \mathcal{V}_{0hb}$ with $\mathbf{W}_h \in \mathcal{V}_{0h}$, $\mathbf{W}_b \in \mathcal{V}_b$ and the amount of artificial viscosity, $\delta_h(\mathbf{U}_h)$, is calculated on the element-level by using the YZβ shock-capturing viscosity parameter (Tezduyar and Senga, 2006),

$$\delta_h(\mathbf{U}_h) = \delta_{yz\beta}(\mathbf{U}_h),$$

with $\delta_{yz\beta}$ defined in (2.19).

In Eq. (2.19) the parameter β is set as $\beta = 1$ for smooth gradient regions and $\beta = 2$ for sharp gradient regions. Therefore, the Eq. (2.19) can be rewritten as,

$$\delta_h(\mathbf{U}_h)\Big|_{\beta=1} = \frac{h}{2} \frac{\|\mathbf{Y}^{-1} R(\mathbf{U}_h)\|_2}{\|\mathbf{Y}^{-1}(\nabla \mathbf{U}_h)^T\|_2} \tag{3.8}$$

and

$$\delta_h(\mathbf{U}_h)\Big|_{\beta=2} = \frac{h^2}{4} \frac{\|\mathbf{Y}^{-1} R(\mathbf{U}_h)\|_2}{\|\mathbf{Y}^{-1} \mathbf{U}_h\|_2}. \tag{3.9}$$

The compromise between the $\beta = 1$ and $\beta = 2$ selections, in Eq. (3.7), was defined in (Tezduyar and Senga, 2006; Tezduyar, 2007) as the following average expression for $\delta_h(\mathbf{U}_h)$:

$$\delta_h(\mathbf{U}_h) = \frac{1}{2}\left(\delta_h(\mathbf{U}_h)\Big|_{\beta=1} + \delta_h(\mathbf{U}_h)\Big|_{\beta=2}\right).$$

3.4 The NMV2 method

The main motivation to construct the NMV2 method was the idea of adding different operators of artificial diffusion into macro and micro scales. The nonlinear operator added to resolved scale works like a shock-capturing term used in the classical stabilized formulations, such as SUPG+YZβ and CAU methods. On the unresolved (or fine) scale of the discretization is added a nonlinear dissipative operator, similar to the Nonlinear Subgrid Scale (NSGS) method presented in (Santos and Almeida, 2007).

The NMV2 method for the Euler equation consists of finding $\mathbf{U}_{hb} = \mathbf{U}_h + \mathbf{U}_b \in \mathcal{V}_{Zhb}$ with $\mathbf{U}_h \in \mathcal{V}_{Zh}$, $\mathbf{U}_b \in \mathcal{V}_b$ such that

$$\underbrace{\int_\Omega \mathbf{W}_{hb} \cdot \Big(\frac{\partial \mathbf{U}_{hb}}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_{hb}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_{hb}}{\partial y}\Big) d\Omega}_{\text{Galerkin term}} \; +$$

$$\underbrace{\sum_{e=1}^{nel} \int_{\Omega_e} \delta_b(\mathbf{U}_h) \Big(\frac{\partial \mathbf{W}_b}{\partial x} \cdot \frac{\partial \mathbf{U}_b}{\partial x} + \frac{\partial \mathbf{W}_b}{\partial y} \cdot \frac{\partial \mathbf{U}_b}{\partial y}\Big) d\Omega}_{\text{Nonlinear subgrid stabilization term}} \; +$$

$$\underbrace{\sum_{e=1}^{nel} \int_{\Omega_e} \delta_h(\mathbf{U}_h) \Big(\frac{\partial \mathbf{W}_h}{\partial x} \cdot \frac{\partial \mathbf{U}_h}{\partial x} + \frac{\partial \mathbf{W}_h}{\partial y} \cdot \frac{\partial \mathbf{U}_h}{\partial y}\Big) d\Omega}_{\text{Shock-capturing term on } \nu_{Zh}} \; = \; \mathbf{0}, \quad \forall \; \mathbf{W}_{hb} \in \mathcal{V}_{0hb}, \tag{3.10}$$

where $\mathbf{W}_{hb} = \mathbf{W}_h + \mathbf{W}_b \in \mathcal{V}_{0hb}$ with $\mathbf{W}_h \in \mathcal{V}_{0h}$, $\mathbf{W}_b \in \mathcal{V}_b$.

The amount of fine-scale artificial viscosity is defined on the element-level by YZβ for smooth gradient regions (i.e., $\beta = 1$)

$$\delta_b(\mathbf{U}_h) = \frac{h}{2} \frac{\|\mathbf{Y}^{-1} R(\mathbf{U}_h)\|_2}{\|\mathbf{Y}^{-1} (\nabla \mathbf{U}_h)^T\|_2}, \tag{3.11}$$

where $R(\mathbf{U}_h)$ is the residue of resolved solution on Ω_e (Eq. (2.14)) and $\mathbf{Y}$ is a diagonal matrix constructed from the reference values of the components of $\mathbf{U}$ (Eq. (2.20)). The local length scale h is defined in Eq. (2.21). An expression similar to (3.11) is proposed in (Santos and Almeida, 2007), based on the concept of minimum kinetic energy in order to measure the quantity of fine-scale artificial dissipation needed in scalar advection-diffusion-reaction problems. But the parameter introduced in (Santos and Almeida, 2007) takes no account of information about the reference values of the problem as in (3.11) via matrix $\mathbf{Y}$ or of the subgrid mesh parameter h defined by Eq. (2.21).

The stabilization parameter of the shock-capturing operator on the resolved scale is a slight modification of the YZβ shock-capturing (Tezduyar and Senga, 2006), and is written as

$$\delta_h(\mathbf{U}_h) = \zeta(M) \left(\frac{h}{2} \frac{\|\mathbf{Y}^{-1} R(\mathbf{U}_h)\|_2}{\|\mathbf{Y}^{-1} (\nabla \mathbf{U}_h)^T\|_2} + \frac{h^2}{4} \frac{\|\mathbf{Y}^{-1} R(\mathbf{U}_h)\|_2}{\|\mathbf{Y}^{-1} \mathbf{U}_h\|_2} \right), \tag{3.12}$$

where

$$\zeta(M) = \begin{cases} \dfrac{M}{4}, & \text{if } M > 2; \\ \\ \dfrac{1}{2}, & \text{otherwise} \end{cases}$$

is a parameter used to adjust the numerical dissipation according to the Mach number, denoted by M. As stated by Tezduyar (2001), excessive numerical dissipation is not always easy to detect. This concern makes it desirable to seek and employ stabilized formulations

developed with objectives that include keeping numerical dissipation to a minimum. Since the goal of the original YZβ shock-capturing parameter is to add minimum numerical dissipation, the expression (3.12) introduce a small dissipation weighted by the Mach number M. The choice of $\zeta(M)$, as in Tezduyar and Senga (2007) (see Remark 1), is done empirically, in order to insert a little more dissipation in problems with inflow Mach number greater than 2.

The local length h (Eq. (2.21)) is defined automatically, accounting for the directions of high gradients and the spatial discretization domain. It is worth noting that the stabilized terms in Eq. (3.10) are meant to be considered in regions of the domain where the residual of the equation is relevant.

Remark 1 *Tezduyar and Senga (2007) proposed a stabilization parameter for YZβ operator that takes into account the Mach number and shock intensity across the shock, as follows*

$$\delta_{shock} = \delta_{yz\beta}\Big(1 + \Big(\frac{\|\boldsymbol{\nabla}\rho\|_2 h_{yz\beta}}{\rho_{ref}}\Big)^{b_J}\langle M^{1/b_M} - 1\rangle\Big),$$

where M is the Mach number,

$$\langle M^{1/b_M} - 1\rangle = \begin{cases} 0, & M^{1/b_M} \leqslant 1, \\ M^{1/b_M} - 1, & M^{1/b_M} > 1, \end{cases}$$

and the parameters b_J and b_M can each be set to 1 for smoother gradient region and 2 for sharper gradient region.

3.5 Nonlinear iterative procedure and time schemes

The nonlinear iterative procedure is defined as follows: given $\mathbf{U}_{hb}^i$ at iteration i, find $\mathbf{U}_{hb}^{i+1}$ satisfying formulations (3.7) or (3.10). Considering $\delta_b(\mathbf{U}_h)$ as defined in Eq. (3.11) the following damping factor ω is used to improve de convergence:

$$\begin{aligned} \delta_b(\mathbf{U}_h^i) &= \nu^{i+1}; \\ \nu^{i+1} &= \omega\tilde{\nu}^{i+1} + (1-\omega)\nu^i, \text{ with } \omega \in (0,1) \text{ suitably chosen;} \\ \tilde{\nu}^{i+1} &= \begin{cases} \dfrac{h}{2}\dfrac{\|\mathbf{Y}^{-1}R(\mathbf{U}_h)\|_2}{\|\mathbf{Y}^{-1}(\nabla\mathbf{U}_h)^T\|_2}, \text{ if } \|\mathbf{Y}^{-1}(\nabla\mathbf{U}_h)^T\|_2 > tol_\delta; \\ \\ 0, \text{ otherwise.} \end{cases} \end{aligned} \tag{3.13}$$

The same strategy is applied to determine $\delta_h = \delta_h(\mathbf{U}_h)$, and Eqs. (2.19) and (3.12). The nonlinear convergence is checked for the resolved solution using the relative error $\left(\frac{||U_h^{i+1} - U_h^i||_2}{||U_h^{i+1}||_2}\right)$ for a prescribed tolerance tol_i and a maximum number of nonlinear iterations. In (Tezduyar and Senga, 2006; Tezduyar and Senga, 2007) the number of nonlinear iteration

is fixed in 3. In our simulations, in most situations, the nonlinear convergence occurs with approximately 5 iterations when $tol_i = 10^{-2}$, not presenting substantial gains beyond 3 iterations. Therefore, in our experiments we will use, in general, 3 nonlinear iterations.

The formulation (3.10) can be partitioned in two subproblems, one related to the resolved scale, given by

$$\begin{aligned}
&\int_\Omega \mathbf{W}_h \cdot \Big(\frac{\partial \mathbf{U}_h^{i+1}}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_h^{i+1}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_h^{i+1}}{\partial y}\Big) d\Omega \quad + \\
&\int_\Omega \mathbf{W}_h \cdot \Big(\frac{\partial \mathbf{U}_b^{i+1}}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_b^{i+1}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_b^{i+1}}{\partial y}\Big) d\Omega \quad + \\
&\sum_{e=1}^{nel} \int_{\Omega_e} \delta_*^i \Big(\frac{\partial \mathbf{W}_h}{\partial x} \cdot \frac{\partial \mathbf{U}_h^{i+1}}{\partial x} + \frac{\partial \mathbf{W}_h}{\partial y} \cdot \frac{\partial \mathbf{U}_h^{i+1}}{\partial y}\Big) d\Omega \quad = \quad \mathbf{0}, \quad \forall\ \mathbf{W}_h \in \mathcal{V}_h,
\end{aligned} \tag{3.14}$$

and the other, representing the subgrid scale, is written as

$$\begin{aligned}
&\int_\Omega \mathbf{W}_b \cdot \frac{\partial \mathbf{U}_b^{i+1}}{\partial t} d\Omega + \int_\Omega \mathbf{W}_b \cdot \Big(\frac{\partial \mathbf{U}_h^{i+1}}{\partial t} + \mathbf{A}_x^h \frac{\partial \mathbf{U}_h^{i+1}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_h^{i+1}}{\partial y}\Big) d\Omega \quad + \\
&\sum_{e=1}^{nel} \int_{\Omega_e} \delta_{**}^i \Big(\frac{\partial \mathbf{W}_b}{\partial x} \cdot \frac{\partial \mathbf{U}_b^{i+1}}{\partial x} + \frac{\partial \mathbf{W}_b}{\partial y} \cdot \frac{\partial \mathbf{U}_b^{i+1}}{\partial y}\Big) d\Omega = \mathbf{0}, \quad \forall\ \mathbf{W}_b \in \mathcal{V}_b.
\end{aligned} \tag{3.15}$$

For the NMV1 method the δ_* and δ_{**} are equal to the δ_h defined by Eq. (2.19) and for the NMV2 method $\delta_* = \delta_h$ (Eq. (3.12)) and $\delta_{**} = \delta_b$ (Eq. (3.11)).

Applying the finite element approximation on Eqs. (3.14) and (3.15), and considering (for simplicity) homogeneous Dirichlet boundary conditions, that is, $\mathbf{Z} = \mathbf{0}$ on Γ_D, we arrive at a local system of differential algebraic equations,

$$\begin{bmatrix} \boldsymbol{M}_{hh} & \boldsymbol{M}_{hb} \\ \boldsymbol{M}_{bh} & \boldsymbol{M}_{bb} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{U}}_h \\ \dot{\boldsymbol{U}}_b \end{bmatrix} + \begin{bmatrix} \boldsymbol{K}_{hh} & \boldsymbol{K}_{hb} \\ \boldsymbol{K}_{bh} & \boldsymbol{K}_{bb} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_h \\ \boldsymbol{U}_b \end{bmatrix} = \begin{bmatrix} \mathbf{0}_h \\ \mathbf{0}_b \end{bmatrix}, \tag{3.16}$$

where $\boldsymbol{U}_h$ and $\boldsymbol{U}_b$ are the unknowns on each element Ω_e, whereas $\dot{\boldsymbol{U}}_h$ and $\dot{\boldsymbol{U}}_b$ are their time derivatives. The local matrices on Eq. (3.16) are defined from Eqs. (3.14), and (3.15) where

$$\boldsymbol{M}_{hh} \;:\; \int_{\Omega_e} \mathbf{W}_h \cdot \frac{\partial \mathbf{U}_h^{i+1}}{\partial t} d\Omega; \tag{3.17}$$

$$\boldsymbol{M}_{hb} \;:\; \int_{\Omega_e} \mathbf{W}_h \cdot \frac{\partial \mathbf{U}_b^{i+1}}{\partial t} d\Omega; \tag{3.18}$$

$$\boldsymbol{M}_{bh} \;:\; \int_{\Omega_e} \mathbf{W}_b \cdot \frac{\partial \mathbf{U}_h^{i+1}}{\partial t} d\Omega; \tag{3.19}$$

$$\boldsymbol{M}_{bb} \;:\; \int_{\Omega_e} \mathbf{W}_b \cdot \frac{\partial \mathbf{U}_b^{i+1}}{\partial t} d\Omega; \tag{3.20}$$

$$\boldsymbol{M}_{hh} \;:\; \int_{\Omega_e} \mathbf{W}_h \cdot \Big(\mathbf{A}_x^h \frac{\partial \mathbf{U}_h^{i+1}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_h^{i+1}}{\partial y} \Big) d\Omega \tag{3.21}$$

$$+ \int_{\Omega_e} \delta_*(\mathbf{U}_h^i) \Big(\frac{\partial \mathbf{W}_h}{\partial x} \cdot \frac{\partial \mathbf{U}_h^{i+1}}{\partial x} + \frac{\partial \mathbf{W}_h}{\partial y} \cdot \frac{\partial \mathbf{U}_h^{i+1}}{\partial y} \Big) d\Omega; \tag{3.22}$$

$$\boldsymbol{K}_{hb} \;:\; \int_{\Omega_e} \mathbf{W}_h \cdot \Big(\mathbf{A}_x^h \frac{\partial \mathbf{U}_b^{i+1}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_b^{i+1}}{\partial y} \Big) d\Omega; \tag{3.23}$$

$$\boldsymbol{K}_{bh} \;:\; \int_{\Omega_e} \mathbf{W}_b \cdot \Big(\mathbf{A}_x^h \frac{\partial \mathbf{U}_h^{i+1}}{\partial x} + \mathbf{A}_y^h \frac{\partial \mathbf{U}_h^{i+1}}{\partial y} \Big) d\Omega; \tag{3.24}$$

$$\boldsymbol{K}_{bb} \;:\; \int_{\Omega_e} \delta_{**}(\mathbf{U}_h^i) \Big(\frac{\partial \mathbf{W}_b}{\partial x} \cdot \frac{\partial \mathbf{U}_b^{i+1}}{\partial x} + \frac{\partial \mathbf{W}_b}{\partial y} \cdot \frac{\partial \mathbf{U}_b^{i+1}}{\partial y} \Big) d\Omega. \tag{3.25}$$

The numerical solution is advanced in time by the predictor-corrector algorithm given in (Hughes and Tezduyar, 1984) and adapted for the multiscale framework in (Sedano et al., 2015; Bento et al., 2016) for the Euler equations.

From Eq. (3.16) we have

$$\boldsymbol{M}_{hh}\dot{\boldsymbol{U}}_h^{n+1} + \boldsymbol{M}_{hb}\dot{\boldsymbol{U}}_b^{n+1} + \boldsymbol{K}_{hh}\boldsymbol{U}_h^{n+1} + \boldsymbol{K}_{hb}\boldsymbol{U}_b^{n+1} = \mathbf{0}_h, \tag{3.26}$$

$$\boldsymbol{M}_{bh}\dot{\boldsymbol{U}}_h^{n+1} + \boldsymbol{M}_{bb}\dot{\boldsymbol{U}}_b^{n+1} + \boldsymbol{K}_{bh}\boldsymbol{U}_h^{n+1} + \boldsymbol{K}_{bb}\boldsymbol{U}_b^{n+1} = \mathbf{0}_b. \tag{3.27}$$

By plugging the expression of the α-method $\Big(\boldsymbol{U}^{n+1} = \boldsymbol{U}^n + (1-\alpha)\Delta t \dot{\boldsymbol{U}}^n + \alpha \Delta t \Delta \dot{\boldsymbol{U}}^{n+1} \Big)$ for both scales, into Eq. (3.26) and (3.27), after arranging the terms applying a strategy like Schur complement, we arrive at the system

$$\boldsymbol{M}^* \dot{\mathbf{U}}_h = \boldsymbol{F}^*,$$

where

$$\boldsymbol{M}^* = \boldsymbol{M}_1 - \boldsymbol{N}_1 \boldsymbol{N}_2^{-1} \boldsymbol{M}_2 \qquad \text{and} \qquad \boldsymbol{F}^* = \boldsymbol{R}_1 - \boldsymbol{N}_1 \boldsymbol{N}_2^{-1} \boldsymbol{R}_2,$$

with

$$\begin{aligned}
\boldsymbol{M}_1 &= \boldsymbol{M}_{hh} + \alpha\Delta t\boldsymbol{K}_{hh};\\
\boldsymbol{N}_1 &= \boldsymbol{M}_{hb} + \alpha\Delta t\boldsymbol{K}_{hb};\\
\boldsymbol{M}_2 &= \boldsymbol{M}_{bh} + \alpha\Delta t\boldsymbol{K}_{bh};\\
\boldsymbol{N}_2 &= \boldsymbol{M}_{bb} + \alpha\Delta t\boldsymbol{K}_{bb};\\
\boldsymbol{R}_1 &= -(\boldsymbol{M}_{hh}\dot{\mathbf{U}}_h + \boldsymbol{M}_{hb}\dot{\mathbf{U}}_b) - (\boldsymbol{K}_{hh}\mathbf{U}_h + \boldsymbol{K}_{hb}\mathbf{U}_B);\\
\boldsymbol{R}_2 &= -(\boldsymbol{M}_{bh}\dot{\mathbf{U}}_h + \boldsymbol{M}_{bb}\dot{\mathbf{U}}_b) - (\boldsymbol{K}_{bh}\mathbf{U}_h + \boldsymbol{K}_{bb}\mathbf{U}_b).
\end{aligned}$$

The problems marching in time from initial instant up to a specified final time according to the benchmark problems, i.e., the time evolution occurs in discrete time steps,

$$t_n = n\Delta t, \quad n = 0, \ldots, m \quad \text{with} \quad m = \frac{t_f}{\Delta t},$$

where Δt is a constant step and t_f is the final time.

Algorithm 1 shows the predictor-corrector, considering the same order approximations in time for the micro and macro scales subproblems, where Δt is the time-step; subscripts $n+1$ and n mean the solution on the time-step $n+1$ and n; α is the time advancing parameter; i is the iteration index; tol_i is the nonlinear correction tolerance; and N_2 is a nonsingular diagonal matrix that comes from Eqs. (3.20) and (3.25). In Algorithm 1, the lines 5-10 shows the prediction phase and in lines 12-20 we have the correction phase. The degrees of freedom related to the subgrid space are locally eliminated in favor of the ones of the macro space using an approach like Schur complement. The resulting linear systems of equations are solved by the GMRES method with block diagonal preconditioner and considering all matrices stored by the well-known element-by-element strategy (Hughes and Tezduyar, 1984).

3.6 Numerical results

In this section we present the numerical experiments considering a couple of well-known benchmark transonic and supersonic problems: “Sod’s shock tube”, “oblique shock”, “reflected shock”, “blast wave/explosion”, and “wind tunnel”, discretized by unstructured triangular meshes using Delaunay triangulation through the software Gmsh (Geuzaine and Remacle, 2009). In all problems we use the GMRES solver with 30 vectors to restart, tolerance equal to 10^{-5}, and tol_δ in Eq. (2.26) and (3.13) is set equal to 10^{-8}. After some numerical experiments with different time-steps size, we noticed that there is no significant differences in the solutions, so we set the time-step equals 10^{-3} for all experiments. We compare the new formulations with the SUPG + YZβ and the CAU methods. The tests are performed on a dedicated machine and the total CPU time (to perform pre-processing,

Algorithm 1 Predictor-Corrector algorithm (PC-std)

input: U_h^0, $\dot{U}_h^0$, U_b^0, and $\dot{U}_b^0$
$t \leftarrow 0$
$n \leftarrow 0$
repeat
  $t \leftarrow t + \Delta t$
  $U_h^{n+1,0} \leftarrow U_h^n + (1-\alpha)\Delta t \dot{U}_h^n$
  $\dot{U}_h^{n+1,0} \leftarrow 0$
  $U_b^{n+1,0} \leftarrow U_b^n + (1-\alpha)\Delta t \dot{U}_b^n$
  $\dot{U}_b^{n+1,0} \leftarrow 0$
  $i \leftarrow 0$
  repeat
    $\boldsymbol{R}_1^{n+1,i} \leftarrow -\left(\boldsymbol{M}_{hh}\dot{U}_h^{n+1,i} + \boldsymbol{M}_{hb}\dot{U}_b^{n+1,i}\right) - \left(\boldsymbol{K}_{hh}U_h^{n+1,i} + \boldsymbol{K}_{hb}U_b^{n+1,i}\right)$
    $\boldsymbol{R}_2^{n+1,i} \leftarrow -\left(\boldsymbol{M}_{bh}\dot{U}_h^{n+1,i} + \boldsymbol{M}_{bb}\dot{U}_b^{n+1,i}\right) - \left(\boldsymbol{K}_{bh}U_h^{n+1,i} + \boldsymbol{K}_{bb}U_b^{n+1,i}\right)$
    $\boldsymbol{M}^* \Delta \dot{U}_h^{n+1,i+1} = \boldsymbol{F}^*$
    $U_h^{n+1,i+1} \leftarrow U_h^{n,i} + \alpha \Delta t \Delta \dot{U}_h^{n+1,i+1}$
    $\dot{U}_h^{n+1,i+1} \leftarrow \dot{U}_h^{n+1,i} + \Delta \dot{U}_h^{n+1,i+1}$
    $U_b^{n+1,i+1} \leftarrow U_b^{n,i} + \alpha \Delta t \Delta \dot{U}_b^{n+1,i+1}$
    $\Delta \dot{U}_b^{n+1,i+1} \leftarrow \boldsymbol{N}_2^{-1}\left(\boldsymbol{R}_2^{n+1,i} - \boldsymbol{M}_2 \Delta \dot{U}_h^{n+1,i+1}\right)$
    $\dot{U}_b^{n+1,i+1} \leftarrow \dot{U}_b^{n+1,i} + \Delta \dot{U}_b^{n+1,i+1}$
    $i \leftarrow i + 1$
  until $\frac{||U_h^{n+1,i+1} - U_h^{n+1,i}||_2}{||U_h^{n+1,i+1}||_2} < tol_i$ **or** $i > i_{\text{MAX}}$
  $n \leftarrow n + 1$
until $t < t_f$ ▷ t_f: final time

processing, and post-processing) is calculated by the arithmetic mean after repeating the experiments three times. A machine with the following configuration is used: Intel Core i7-4770 (3.4 GHz) processor; 16GB of RAM memory; and Ubuntu 14.04 operating system. The applications are written in C language and compiled with the GNU gcc-4.8.4 using optimization flags -Ofast -march=native.

3.6.1 Shock tube problem

This problem was originally proposed by Sod (1978). It is a transient fluid flow that has analytic solution. It consists of a one-dimensional tube with two different properties of gases separated by a diaphragm in the middle. The gas to the left and right of the diaphragm is initially at rest. The pressure and density are discontinuous across the diaphragm. At $t = 0$, the diaphragm is broken bringing on a shock wave that propagates from the left side to the right side. The computational domain is the rectangle $\Omega = [0, 1] \times [0, 0.02]$

(Fig. 3.2). The initial conditions on the left and right sides of the tube are given by

$$\text{left} \begin{cases} \rho &= 1.0 \\ u &= 0.0 \\ v &= 0.0 \\ p &= 1.0 \end{cases} \qquad \text{right} \begin{cases} \rho &= 0.125 \\ u &= 0.0 \\ v &= 0.0 \\ p &= 0.1 \end{cases} .$$

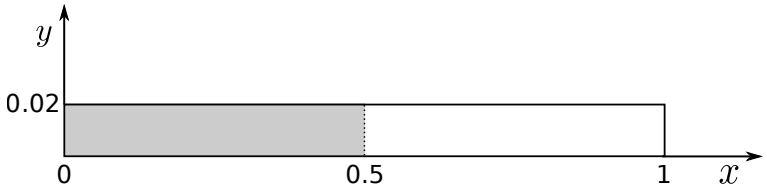


Figure 3.2 – Shock tube problem description

This leads to a transonic flow with maximum Mach number $M = 0.9$ approximately. In this example, we consider a structured mesh tilted to the right with 303 nodes and 400 elements (Fig. 3.3), the simulation runs until $t_f = 0.2$ (200 steps) . For the reference values used in Eq. (2.20), we consider the initial condition values for the left domain.

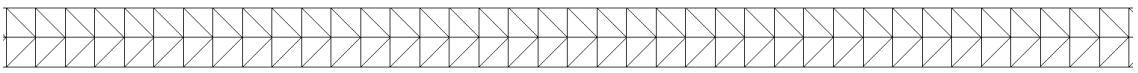


Figure 3.3 – Detail of the shock tube mesh.

Figure 3.4 shows the density profile along $y = 0.01$, obtained with CAU, SUPG + YZβ, NMV1 and NMV2 methods, considering only 3 nonlinear corrections. The solutions obtained with the NMV1 and NMV2 are more accurate than the SUPG + YZβ. The solution obtained with the CAU method presents a little more dissipation. Figure 3.5 shows the density profile along $y = 0.01$ with the nonlinear correction tolerance $tol_i = 10^{-2}$ (measured with the Euclidean norm) and the maximum number of nonlinear corrections equals 10. In this case we may observe some oscillations in the solutions obtained with the SUPG + YZβ method.

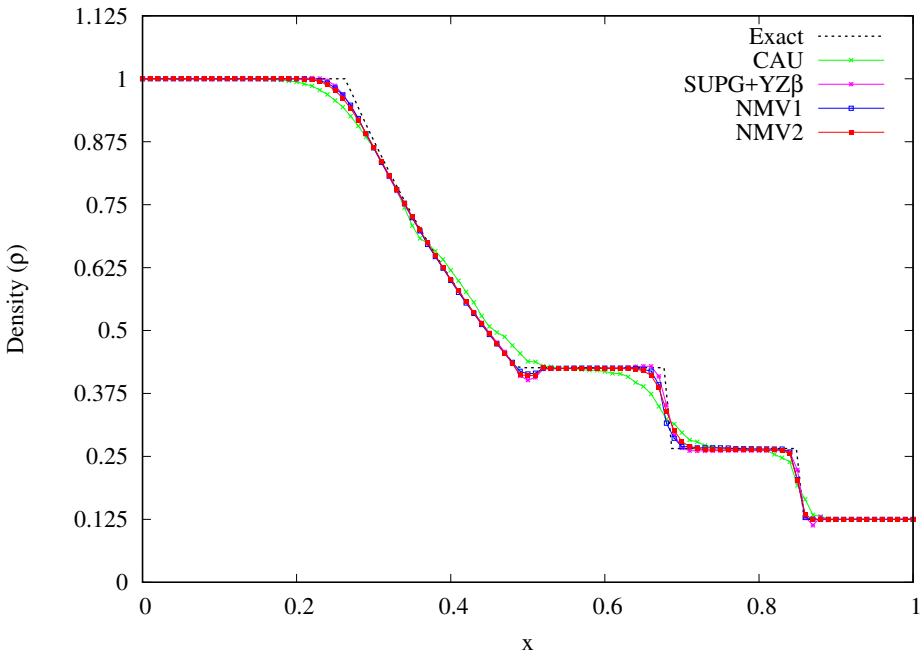


Figure 3.4 – Shock tube problem: Density profile along $y = 0.01$. Iterative procedure: 3 nonlinear corrector steps.

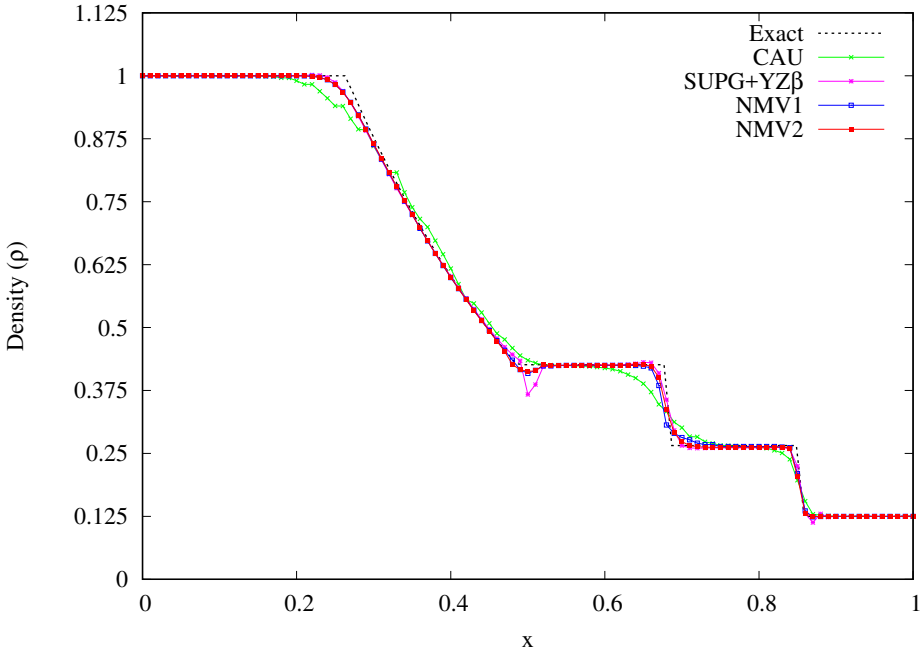


Figure 3.5 – Shock tube problem: Density profile along $y = 0.01$. Iterative procedure: $tol_i = 10^{-2}$ and $i_{\max} = 10$.

As we can see in Table 3.1, the NMV methods considering only 3 nonlinear corrections need fewer GMRES iterations and less CPU time than the CAU and SUPG + YZβ methods. In addition, the NMV1 (NMV2) method requires approximately 76%(87%) and 71%(80%), respectively, of the CPU time required by the CAU and the SUPG + YZβ methods.

Table 3.1 – Computational performance - Shock tube with 3 fixed nonlinear iterations.

Methods	GMRES Iterations	CPU Time (s)
CAU	6,639	0.45193
SUPG+YZβ	8,560	0.48807
NMV1	3,909	0.34536
NMV2	4,610	0.39219

Table 3.2 shows the computational performance taking into account that the nonlinear process goes on until the convergence criteria is satisfied. We can verify that all methods converge on average with fewer than 10 iterations, the only exception is the CAU method that spend approximately 10 iterations in each linearization process. We can observe that the NMV1 and NMV2 methods present similar behavior and a better performance compared with the CAU and SUPG + YZβ methods.

Table 3.2 – Computational performance - Shock tube with nonlinear tolerance of 10^{-2} and maximum number of iterations is equal to 10.

Methods	GMRES Iter.	Nonlinear Iter. (NL)	NL/200	CPU Time (s)
CAU	33,779	1,997	9.985	1.81797
SUPG+YZβ	14,763	1,015	5.075	0.81927
NMV1	8,051	1,219	6.095	0.67233
NMV2	8,996	1,205	6.025	0.66743

Figure 3.6 shows the time evolution of the density residual $L^2(\Omega)$-norm. The residual sequence of the NMV1 and NMV2 methods have the same behavior, remaining approximately constant at the order of 10^{-4}. Also, the SUPG + YZβ and CAU residual sequence have the same behavior remaining approximately constant at the order of 10^{1}. Figure 3.7 shows the time evolution of the density residual $L^2(\Omega)$-norm with the nonlinear correction tolerance 10^{-2} and the maximum number of nonlinear corrections equals 10. As we can see, the results are similar to those shown in Fig. 3.6, but with lower values.

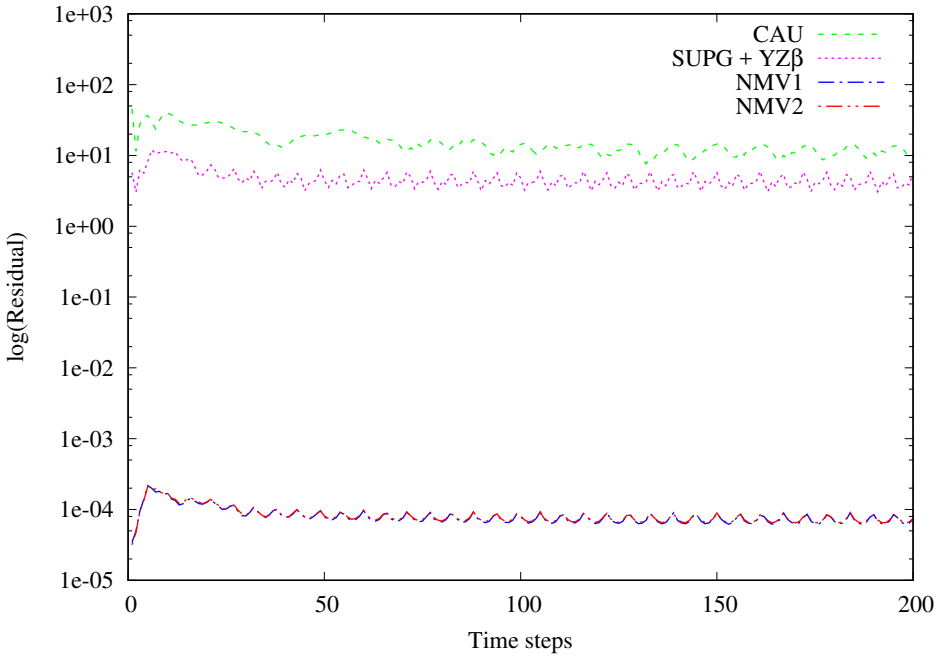


Figure 3.6 – Shock tube problem: Residual of density. Iterative procedure: 3 nonlinear corrector steps.

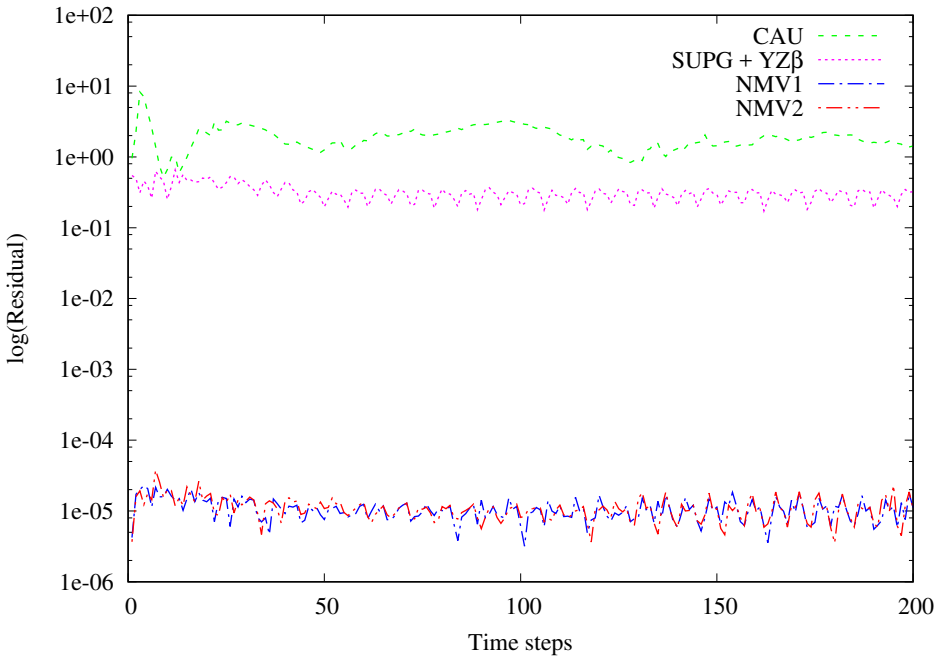


Figure 3.7 – Shock tube problem: Residual of density. Iterative procedure: $tol_i = 10^{-2}$ and $i_{\max} = 10$.

Table 3.3 shows the error in the $L^2(\Omega_y)$ norm, with $\Omega_y = (0,1) \times \{0.01\}$, for density solution of the 1D shock tube problem at time 0.2 and $y = 0.01$. The equation

$$\|\rho(\cdot,t_f) - \rho_h(\cdot,t_f)\|_{L^2(\Omega_y)} = \left(\int_{\Omega_y} \left(\rho(\cdot,t_f) - \rho_h(\cdot,t_f)\right)^2 dx \right)^{\frac{1}{2}},$$

where $t_f = 0.2$ is the final time, shows how the error is calculated from exact and approximated solutions in Fig. 3.4(a). We can verify that NMV methods is slightly more accurate than the SUPG + YZβ and CAU methods.

Table 3.3 – Shock tube: $L^2(\Omega_y)$-norm error, with $\Omega_y = (0,1) \times \{0.01\}$, at time 0.2.

	NMV1	NMV2	YZβ	CAU
$\|\rho - \rho_h\|_{L^2(\Omega_y)}$	1.683335E-02	1.690536E-02	1.710082E-02	2.376853E-02

3.6.2 Oblique shock problem

This problem is a Mach 2 uniform flow over a wedge, at an angle of -10° with respect to a horizontal wall. The solution involves an oblique shock at an angle of 29.3° emanating from the leading edge of the wedge, as shown in Fig. 3.8. The computational domain Ω is a square with $0 \leqslant x \leqslant 1$ and $0 \leqslant y \leqslant 1$. Prescribing the following inflow data on the left and top boundaries results in a solution with the following outflow data:

$$\text{inflow} \begin{cases} M &= 2.0 \\ \rho &= 1.0 \\ u &= \cos 10^0 \\ v &= -\sin 10^0 \\ p &= 0.17857 \end{cases} \qquad \text{outflow} \begin{cases} M &= 1.64052 \\ \rho &= 1.45843 \\ u &= 0.88731 \\ v &= 0.0 \\ p &= 0.30475 \end{cases}.$$

Four Dirichlet boundary conditions are imposed at the left and the top boundaries, the reflection condition is set at the bottom boundary, and no boundary condition is imposed at the outflow (right) boundary.

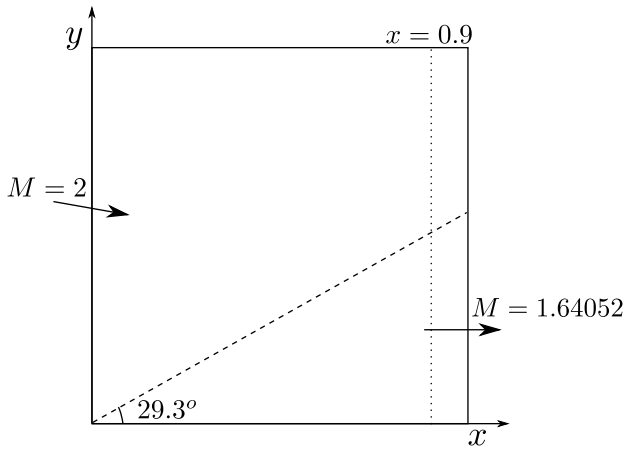


Figure 3.8 – Oblique shock problem description

For all simulations we consider an unstructured mesh consisting of 474 nodes and 874 elements in which all elements have approximately equal areas. For the reference values used in Eq. (2.20), we consider the initial condition values for the left domain. The simulation runs until 3,000 steps with 3 fixed nonlinear iterations. Despite the final time being $t = 3.0$, the numerical solution of each method do not present substantial difference from time $t = 2.0$. Figure 3.9 shows the density profile along $x = 0.9$, obtained with CAU, SUPG + YZβ, NMV1 and NMV2 methods. The solution obtained with the SUPG + YZβ is slightly better than the NMV1 and NMV2 on the left of the shock, whereas the solution with NMV1 and NMV2 are better on the right of the shock. The CAU method clearly exhibits more dissipation. Additionally, the solutions obtained with the NMVs, and the CAU methods are more symmetrical, in relation to the exact solution, than that obtained with the SUPG+YZβ method.

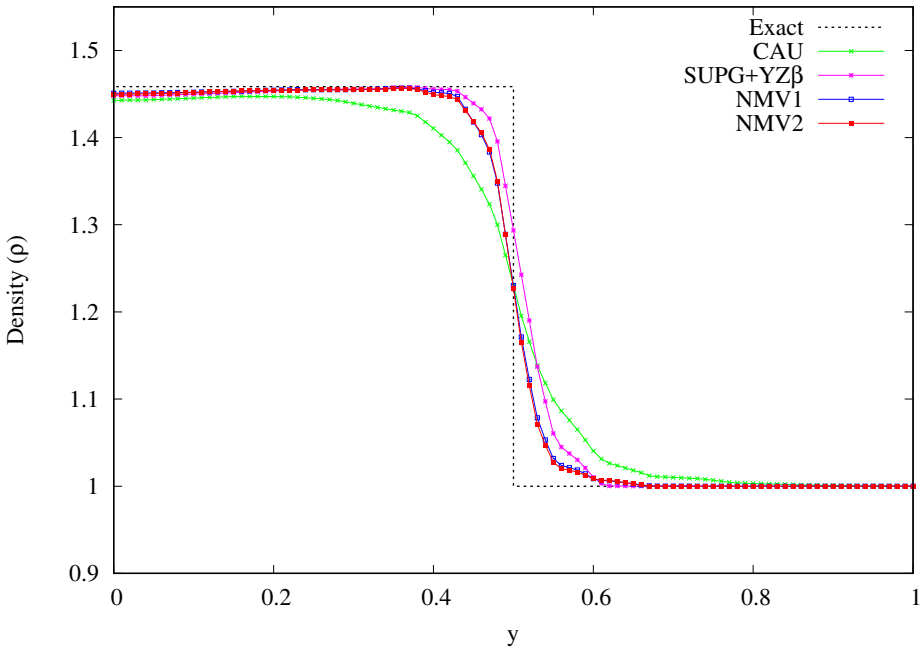


Figure 3.9 – Oblique shock problem: Density profile along $x = 0.9$.

On the other hand, the NMV1 and the NMV2 methods need fewer GMRES iterations and less CPU time than the others, as shown in Table 3.4. Furthermore, the NMV1 (NMV2) method requires approximately 52%(64%) of the CPU time required by the CAU method. Whereas, the NMV1 (NMV2) method required approximately 51%(63%) of the CPU time required by the SUPG + YZβ method. The NMV1 method needs 20% less CPU time than the NMV2 method.

Table 3.4 – Computational performance - Oblique shock.

Methods	GMRES Iterations	CPU Time (s)
CAU	178,695	18.24322
SUPG+YZβ	184,981	18.49676
NMV1	45,038	9.35786
NMV2	47,733	11.72239

Figure 3.10 shows the time evolution of the density residual $L^2(\Omega)$-norm. The NMV1 and NMV2 methods present the same behavior with the residual sequence reaching to approximately 10^{-5} (for $t_f = 3.0$) whereas the SUPG + YZβ residual sequence reaches to approximately 10^{-2} in fewer than 3,000 steps. For the CAU method, the residue remains approximately constant at the order of 10^0 throughout the time evolution.

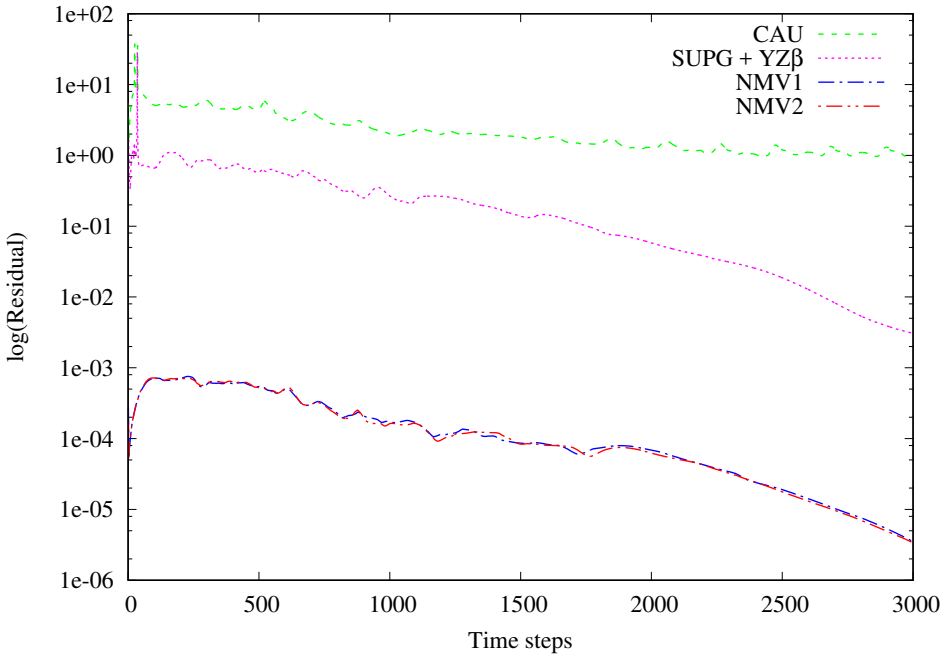


Figure 3.10 – Oblique shock problem: Residual of density.

3.6.3 Reflected shock problem

This problem consists of three regions (①, ②, and ③) separated by an oblique shock and its reflection from a wall, as shown in Fig. 3.11. Prescribing the following Mach 2.9 inflow data in the first region on the left, Mach 2.3781 inflow data in the second region on the top, and requiring the incident shock to be at an angle of 29°, leads to the following

exact solution at the other two regions (② and ③):

$$
\textcircled{1}\begin{cases} M &= 2.9 \\ \rho &= 1.0 \\ u &= 2.9 \\ v &= 0.0 \\ p &= 0.714286 \end{cases}
\quad
\textcircled{2}\begin{cases} M &= 2.3781 \\ \rho &= 1.7 \\ u &= 2.61934 \\ v &= -0.50632 \\ p &= 1.52819 \end{cases}
\quad
\textcircled{3}\begin{cases} M &= 1.94235 \\ \rho &= 2.68728 \\ u &= 2.40140 \\ v &= 0.0 \\ p &= 2.93407 \end{cases}
\tag{3.28}
$$

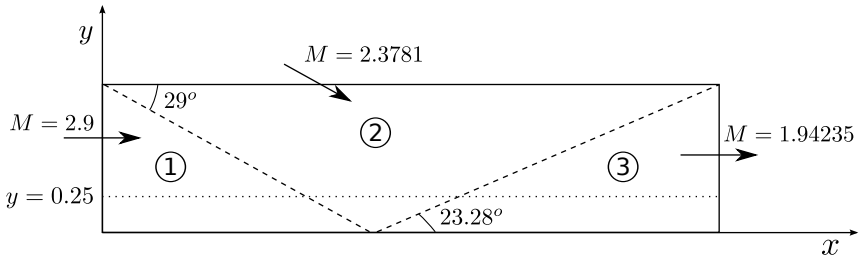


Figure 3.11 – Reflected shock problem description.

The computational domain Ω is a rectangle with $0 \leqslant x \leqslant 4.1$ and $0 \leqslant y \leqslant 1$. We prescribe the density, the velocities and the pressure on the left and top boundaries, the slip condition is imposed at the bottom boundary, and no boundary condition is imposed at the outflow (right) boundary.

For all simulations we consider an unstructured mesh consisting of 1,404 nodes and 2,646 elements in which all elements have approximately equal areas. The reference values used in Eq. (2.20) are the initial condition values for the left domain. The simulation runs until $t_f = 3.0$ (3,000 steps) with 3 fixed nonlinear iterations. Figure 3.12 shows the density profile along $y = 0.25$. The solutions obtained with the NMV1 and the SUPG + YZβ methods are in good agreement, slightly more accurate than the solution obtained with the NMV2 method. The solution presented by the CAU method is more dissipative. On the other hand, looking at the enlarged region, we note that SUPG + YZβ and NMV1 methods experience small under- and over- shoots, respectively, whereas the NMV2 solution is monotone throughout the domain.

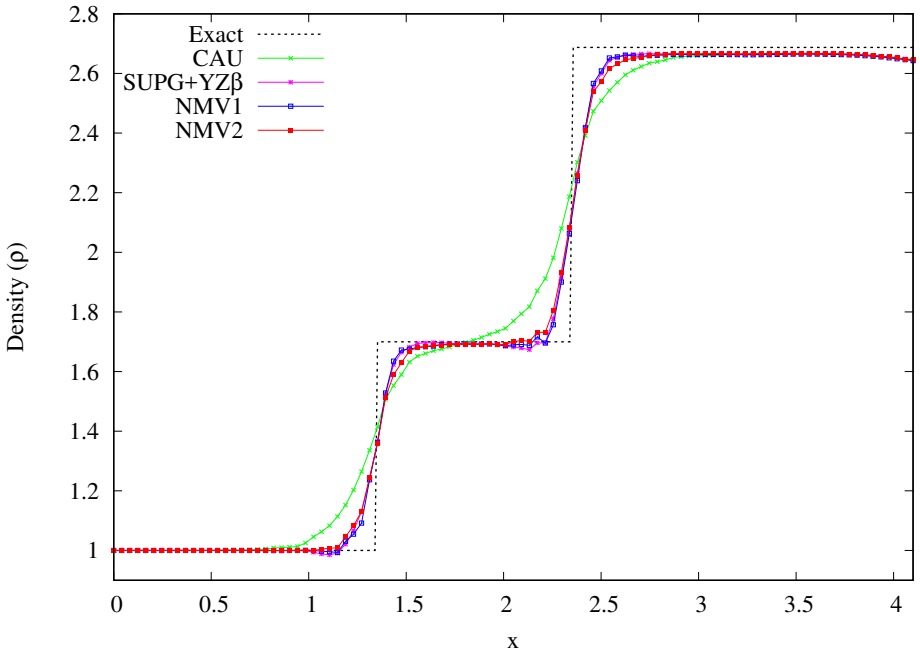


Figure 3.12 – Reflected shock problem: Density profile along $y = 0.25$.

One more time, the NMV1 and NMV2 methods need fewer GMRES iterations and less CPU time than the others, as shown in Table 3.5. Additionally, the NMV1 (NMV2) method requires approximately 46%(58%) of the CPU time required by the CAU method, and approximately 50%(64%) of the CPU time required by the SUPG + YZβ method.

Table 3.5 – Computational performance - Reflected shock.

Methods	GMRES Iterations	CPU Time (s)
CAU	214,601	67.40413
SUPG+YZβ	196,356	61.27936
NMV1	47,603	30.83919
NMV2	56,107	39.18147

Figure 3.13 shows the time evolution of the density residual $L^2(\Omega)$-norm. The residual sequence of the NMV1 and NMV2 methods reach to approximately 10^{-4} in less than 3,000 steps, whereas the SUPG + YZβ method reaches to approximately 10^{-1}. The behavior of the CAU residual sequence is similar to the previous example, remaining approximately constant in the order of 10^1.

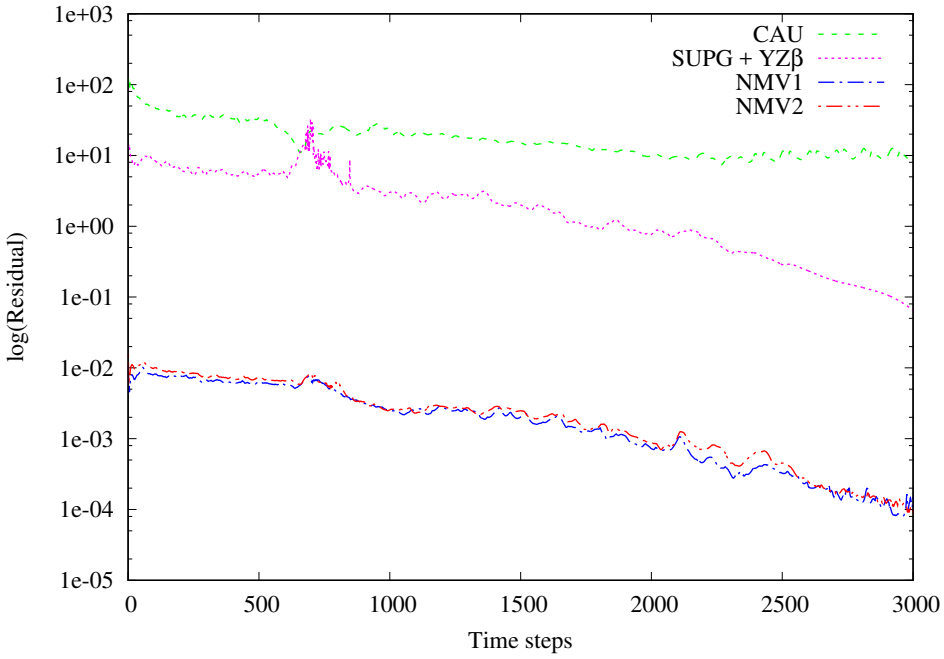


Figure 3.13 – Reflected shock problem: Residual of density.

3.6.4 Explosion problem

We consider the explosion problem as described by Toro (2009). It is a circular symmetric 2D problem on a square domain, $\Omega = [0, 2] \times [0, 2]$. The initial condition consists of a circular region with radius $R = 0.4$ centered at $(1, 1)$ with higher density and higher pressure and the region outside the circle, see Fig. 3.14. The flow variables are constant in each of these regions and are separated by a circular discontinuity at time $t = 0$. The two constant states are chosen as

$$ins \begin{cases} \rho &= 1.0 \\ u &= 0.0 \\ v &= 0.0 \\ p &= 1.0 \end{cases} \qquad out \begin{cases} \rho &= 0.125 \\ u &= 0.0 \\ v &= 0.0 \\ p &= 0.1 \end{cases} \tag{3.29}$$

Subscripts *ins* and *out* denote values inside and outside the shaded circle, respectively.

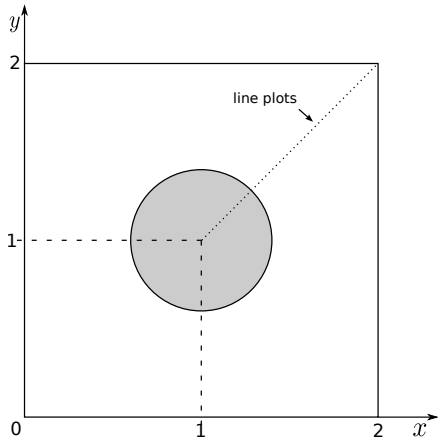


Figure 3.14 – Explosion problem description.

A reference solution is used considering a fine mesh with $1,000 \times 1,000$ computing cells by the Weighted Average Flux (WAF) method (Toro, 2009). Previous works, such as (Toro, 2009; Abbassi et al., 2014), smoothed the initial discontinuity over a few grid points in order to stabilize the simulation. Here, however, we do not use this procedure. In our simulations, we consider an unstructured mesh with 13,470 nodes and 26,538 elements in which all elements have approximately equal areas. As in the shock tube problem (3.6.1), the reference values in Eq. (2.20) are those that generate the movement (the initial condition values inside the circle), producing a shock wave radially. Once the reference solution is given in $t_f = 0.25$, the simulation runs until 250 steps considering $\Delta t = 10^{-3}$, and we use the nonlinear iterations fixed in 3. Figure 3.15 compares the radial variations of the density obtained using CAU, SUPG + YZβ, NMV1 and NMV2 methods. The solutions obtained with NMV1 and NMV2 are visually very similar, but with small differences in relation to the SUPG + YZβ, mainly in the region near $x = 0$ and in the discontinuity near $x = 0.8$, in which the SUPG + YZβ presents a small undershoot. The CAU method is the most dissipative.

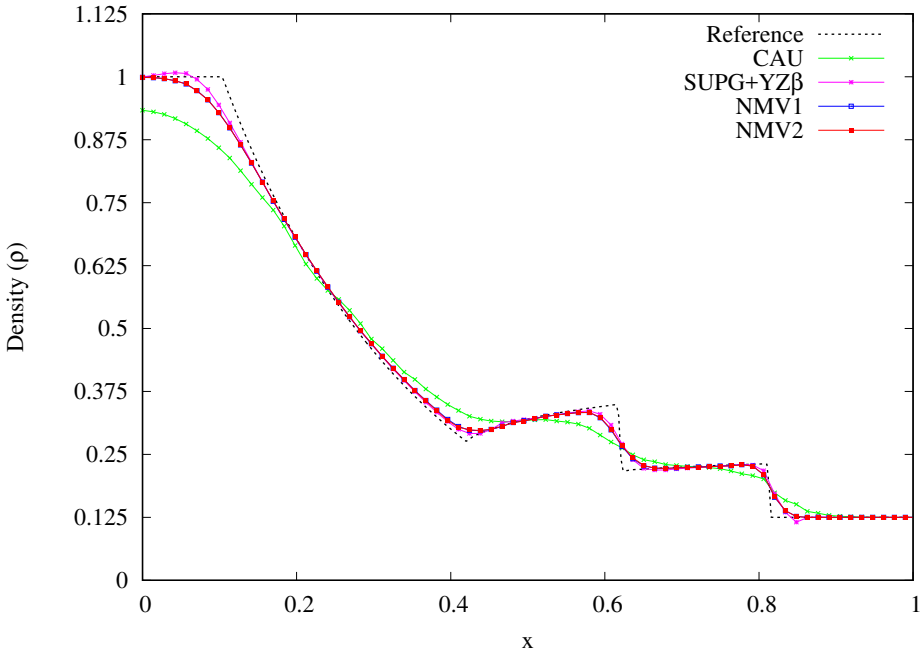


Figure 3.15 – Explosion problem: Comparisons of radial variations of the density field.

Once again, the NMV1 and NMV2 methods need fewer GMRES iterations and less CPU time than the others, as we can see in Table 3.6. In addition, the NMV1 (NMV2) method requires approximately 80%(87%) of the CPU time required by the CAU method, and approximately 74%(80%) of the CPU time required by the SUPG + YZβ method.

Table 3.6 – Computational performance - Explosion.

Methods	GMRES Iterations	CPU Time (s)
CAU	9,271	63.88577
SUPG+YZβ	11,145	69.97707
NMV1	5,685	51.42688
NMV2	5,871	55.66704

Figure 3.16 shows the time evolution of the density residual $L^2(\Omega)$-norm for all methods. The residual sequence of the NMV1 and the NMV2 methods remaining in the vicinity of the value 10^{-3}. The same behavior occur with the SUPG + YZβ and CAU residual sequence, remaining in the vicinity of the value 10^1 and 10^2, respectively.

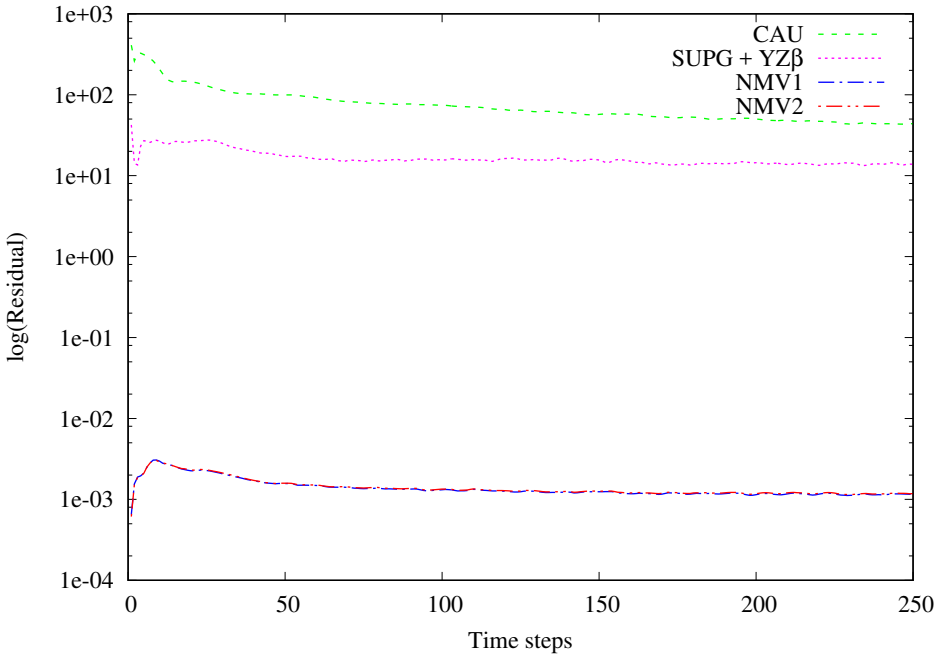


Figure 3.16 – Explosion problem: Residual of density.

3.6.5 Wind tunnel problem

This two-dimensional test problem was originally introduced by Emery (1968) and since then this problem has proven to be a useful test for a large number of methods in fluid dynamics. The computational domain is shown in Fig. 3.17 and the inflow data on the left boundary is set up by

$$inflow \begin{cases} M &= 3.0 \\ \rho &= 1.4 \\ u &= 3.0 \\ v &= 0.0 \\ p &= 1.0 \end{cases}. \tag{3.30}$$

Along the walls of the tunnel reflecting boundary conditions are applied and no boundary condition is imposed at the outflow boundary.

The corner of the step is the center of a rarefaction fan being a singular point of the flow. As pointed out in (Abbassi et al., 2014), the schemes can be modified near the corner in order to stabilize the flow. In our simulations, we do not use any treatment of the flow near the corner, neither mesh refinement. For all simulations we consider an unstructured mesh consisting of 3,805 nodes and 7,340 elements in which all elements have approximately equal areas. The simulation runs until $t_f = 3.0$ (3,000 steps) with the nonlinear correction tolerance 10^{-2} and the maximum number of nonlinear corrections equals 10. For the reference values used in Eq. (2.20), we consider the inflow values (Eq.

(3.30)). The reference solution used to compare it with the methods described here is found in (Abbassi et al., 2014), where a mesh with $14,175$ cubic elements is used.

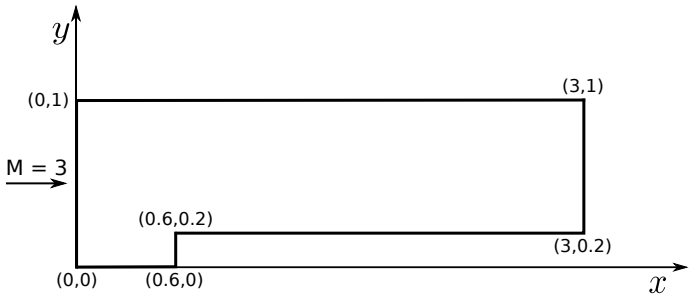
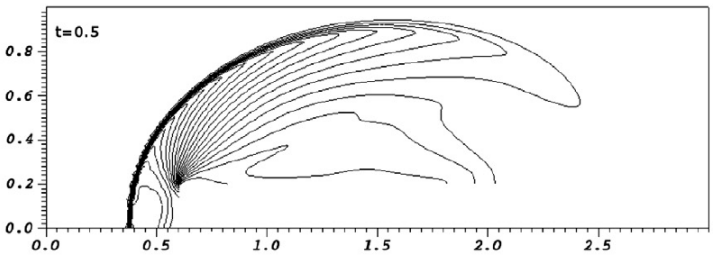
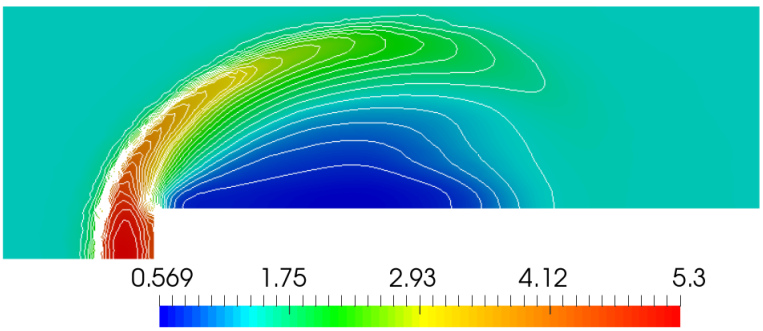


Figure 3.17 – Wind tunnel problem description.

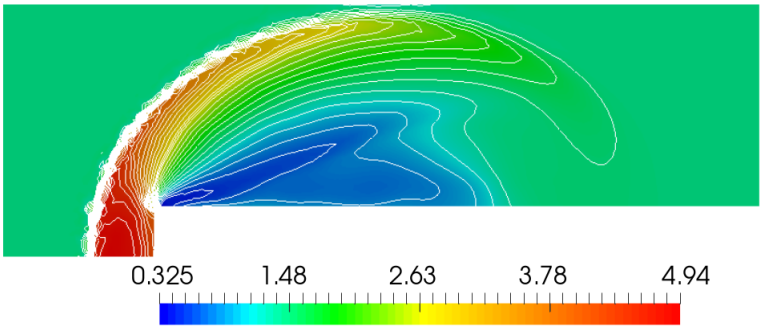
In Fig. 3.18 - 3.23 the 2D density distribution solutions are shown at time intervals of 0.5 up to time 3.0, obtained by the CAU, the SUPG + YZβ, the NMV1, and the NMV2 methods. Fig. 3.18 - 3.20 shows the solutions until time $t = 1,5$. All methods present results visually close to the reference solutions, however, the solutions obtained with the CAU and the NMV2 methods are slightely better. On the other hand, from $t = 2.0$ until $t = 3.0$ (Fig. 3.21 - 3.23), all the methods present different solutions, where the SUPG+YZβ and the NMV1 methods present solutions further away from reference solutions, showing an advanced profile. This seems to be a feature of the SUPG + YZβ method in this problem, as shown in (Catabriga et al., 2009) where the solutions are advanced in comparison to those found in (Woodward and Colella, 1984) and also in the more recent work (Abbassi et al., 2014). Since this problem is in transition phase ($M = 3.0$ at the inflow) from supersonic to hypersonic, it requires more numerical dissipation due to the high velocity. Thus, the most diffusive method in all the previous tests (CAU method) presents the best results (Fig. 3.18 - 3.23), as shown (at the final time $t_f = 3.0$) in Fig. 3.23(b) in comparison with the reference solution (Fig. 3.23(a)). It is worth pointing out that the NMV2 method presents better solutions than the NMV1 method, since it adds more artificial viscosity (on resolved scales) when $M > 2$. This suggests that the NMV2 solutions can be further improved for this kind of problem by correctly setting the Mach dependency of the stabilization parameter. A fast option could be to use the parameter designed by Tezduyar and Senga (2007), described in Remark 1.



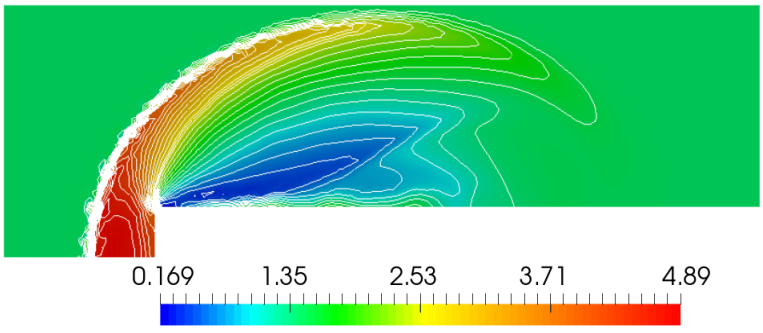
(a) Reference solution - 14,175 cubic elements (Source: Adapted from Abbassi et al. (2014)).



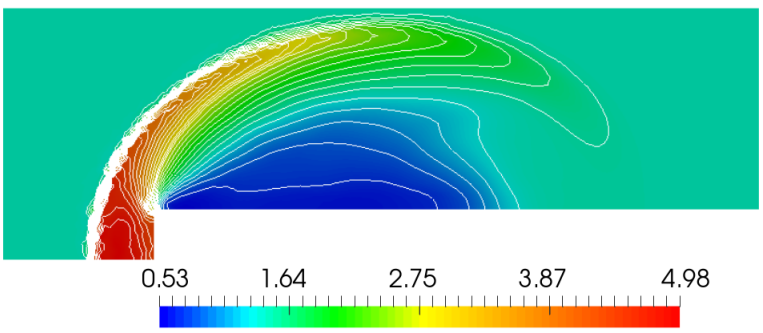
(b) CAU



(c) $SUPG + YZ\beta$

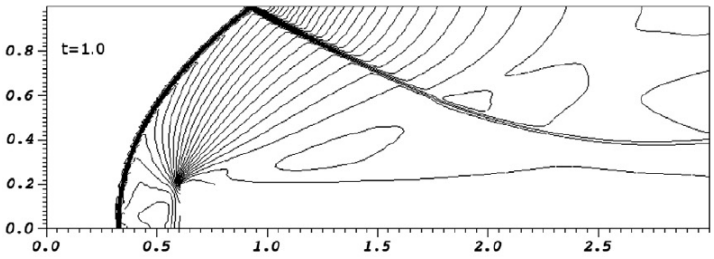


(d) $NMV1$

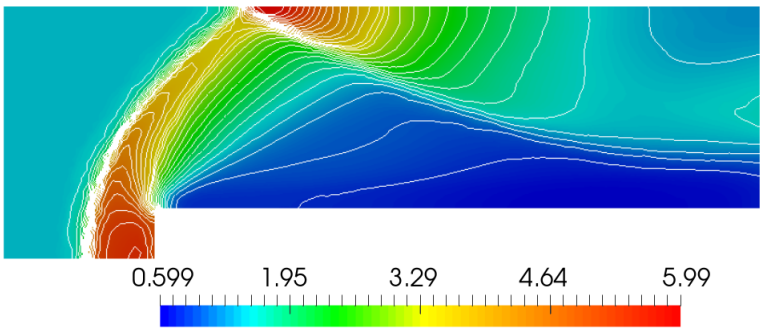


(e) $NMV2$

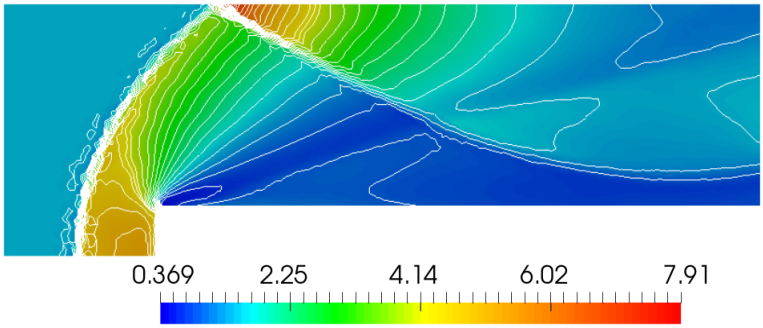
Figure 3.18 – Wind tunnel problem: density distribution 2D solution at time $t = 0.5$.



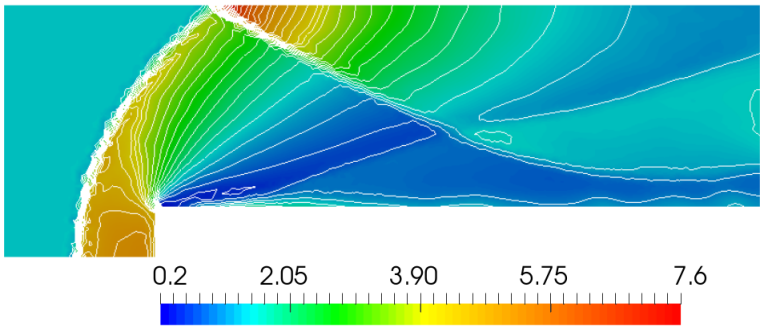
(a) Reference solution - 14,175 cubic elements (Source: Adapted from Abbassi et al. (2014)).



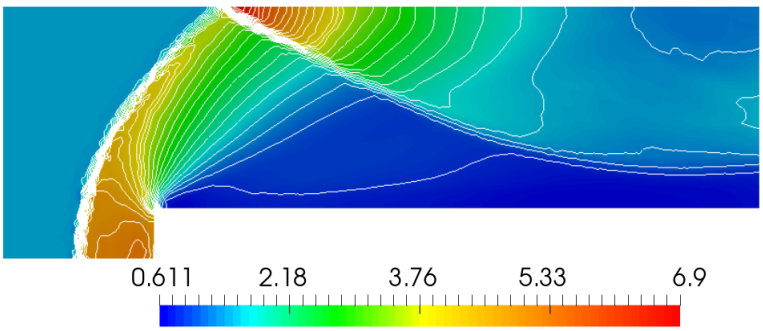
(b) CAU



(c) $SUPG + YZ\beta$

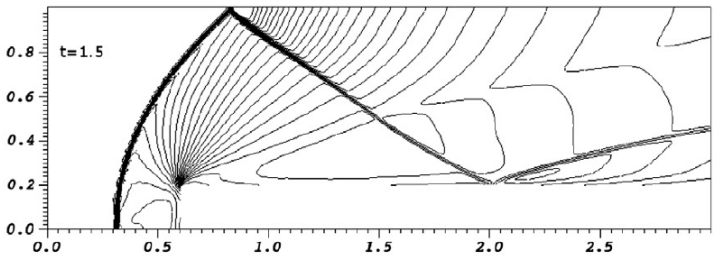


(d) $NMV1$

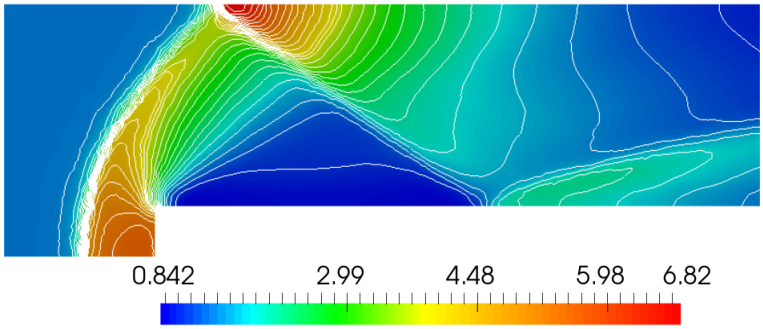


(e) $NMV2$

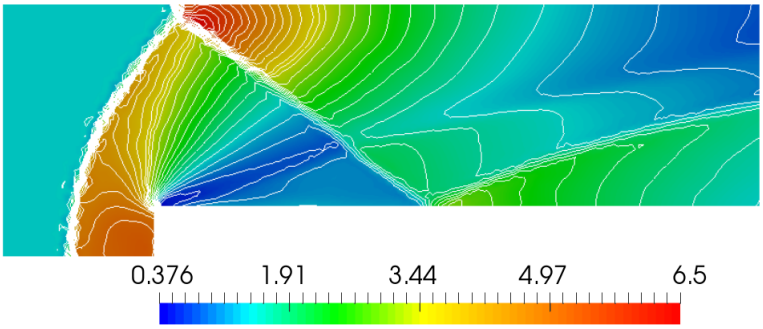
Figure 3.19 – Wind tunnel problem: density distribution 2D solution at time $t = 1.0$.



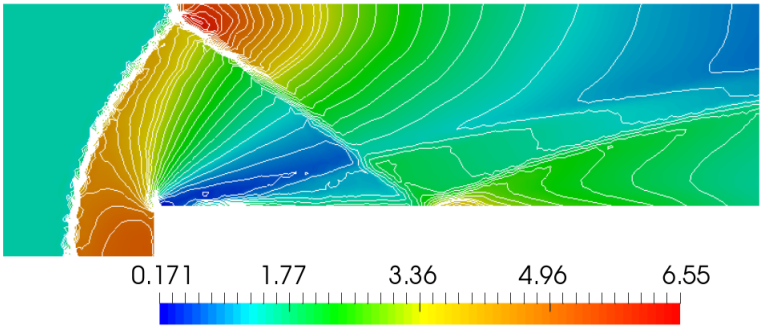
(a) Reference solution - 14,175 cubic elements (Source: Adapted from Abbassi et al. (2014)).



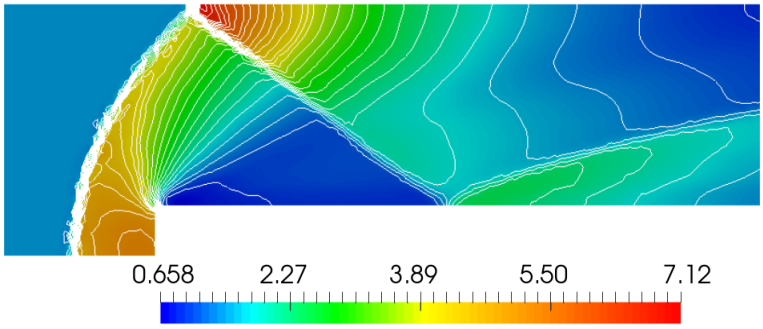
(b) CAU



(c) $SUPG + YZ\beta$

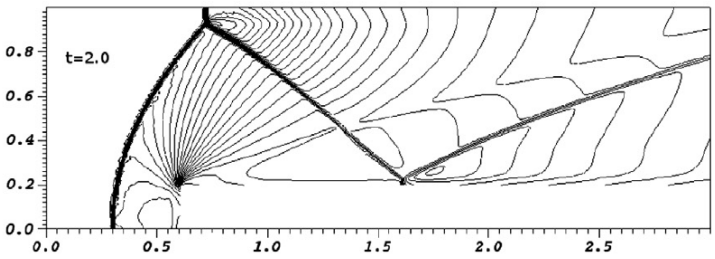


(d) $NMV1$

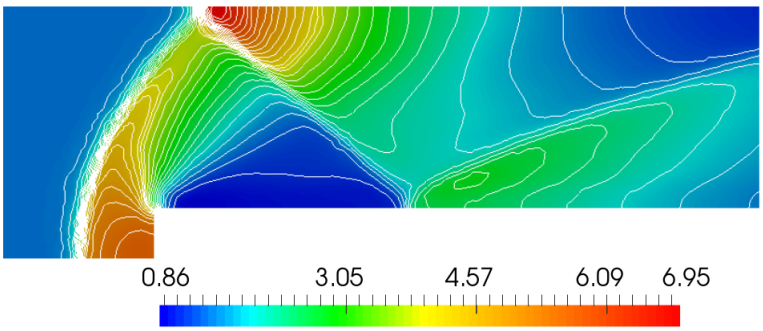


(e) $NMV2$

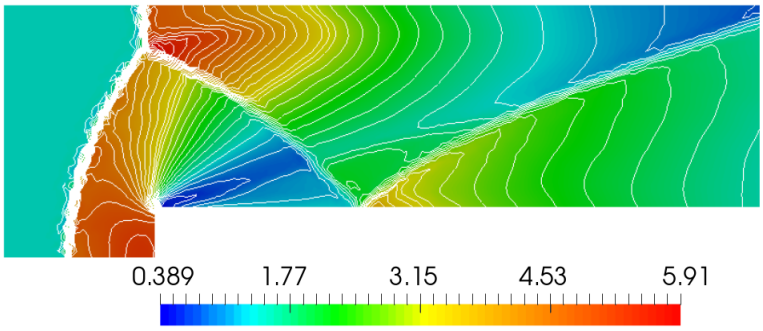
Figure 3.20 – Wind tunnel problem: density distribution 2D solution at time $t = 1.5$.



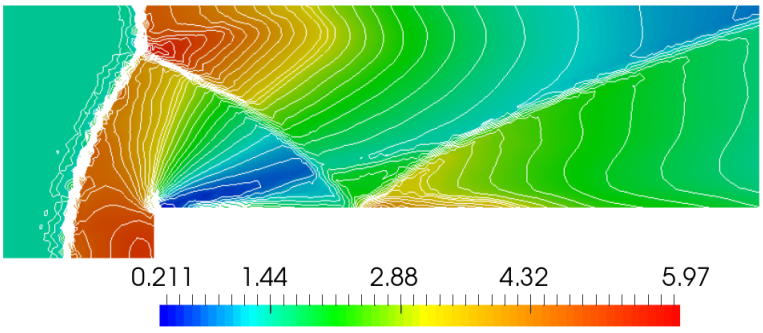
(a) Reference solution - 14,175 cubic elements (Source: Adapted from Abbassi et al. (2014)).



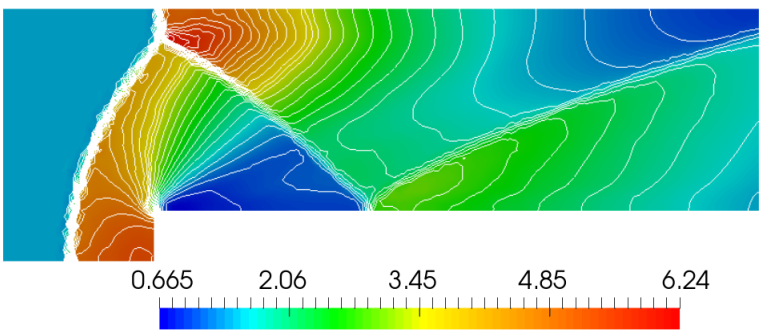
(b) CAU



(c) $SUPG + YZ\beta$

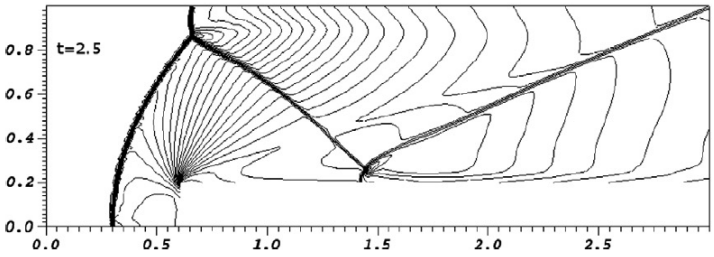


(d) $NMV1$

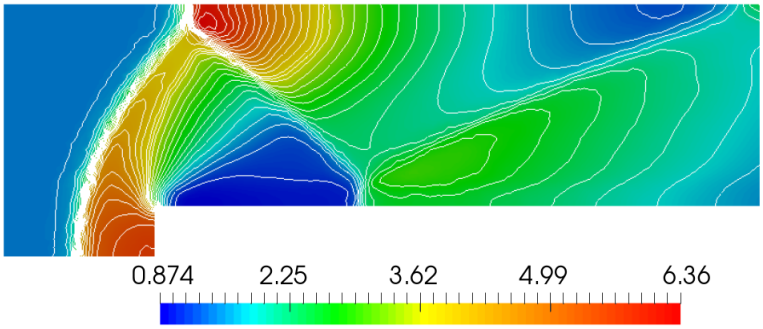


(e) $NMV2$

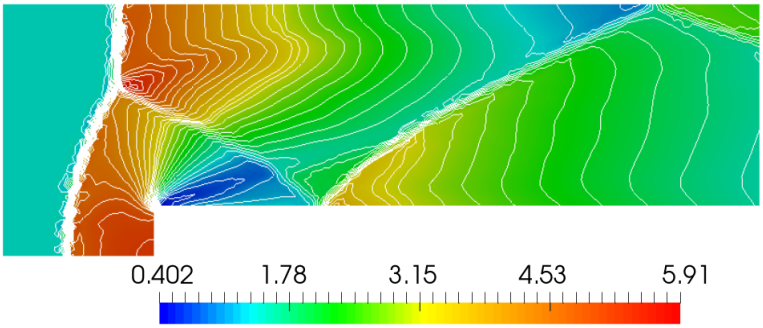
Figure 3.21 – Wind tunnel problem: density distribution 2D solution at time $t = 2.0$.



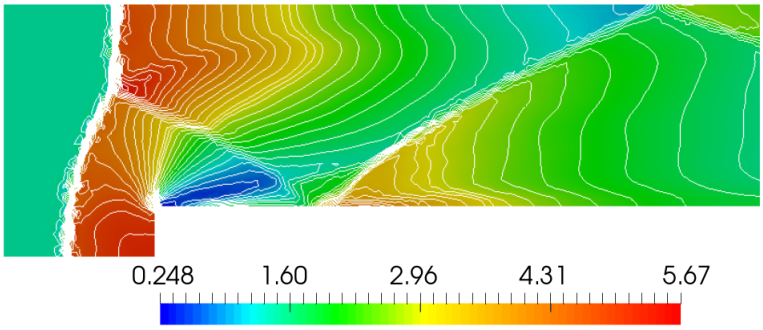
(a) Reference solution - 14,175 cubic elements (Source: Adapted from Abbassi et al. (2014)).



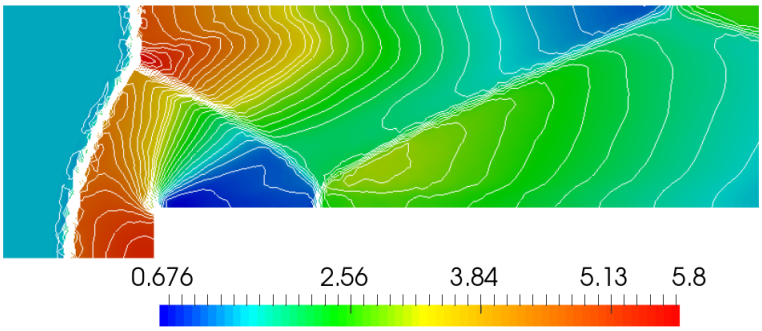
(b) CAU



(c) $SUPG + YZ\beta$

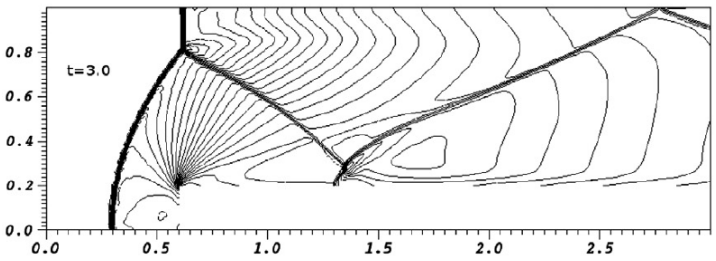


(d) $NMV1$

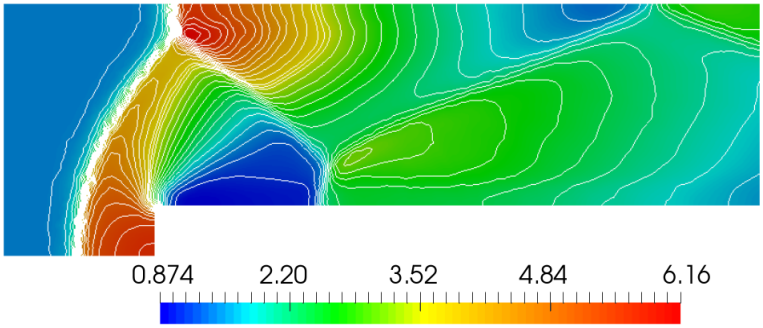


(e) $NMV2$

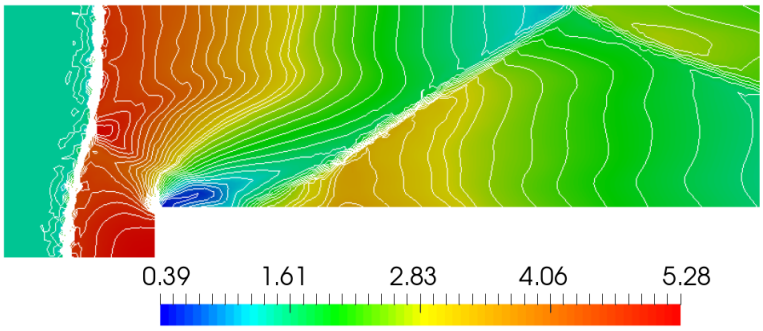
Figure 3.22 – Wind tunnel problem: density distribution 2D solution at time $t = 2.5$.



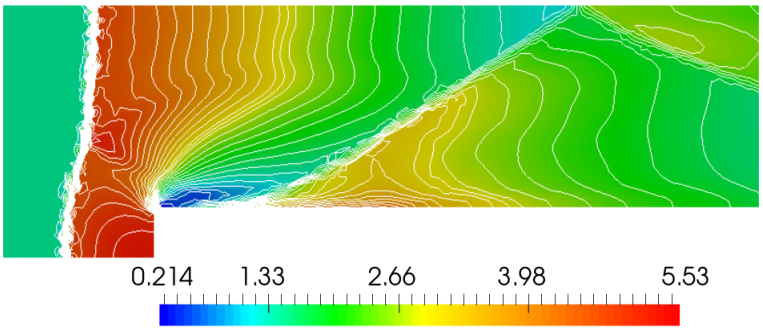
(a) Reference solution - 14,175 cubic elements (Source: Adapted from Abbassi et al. (2014)).



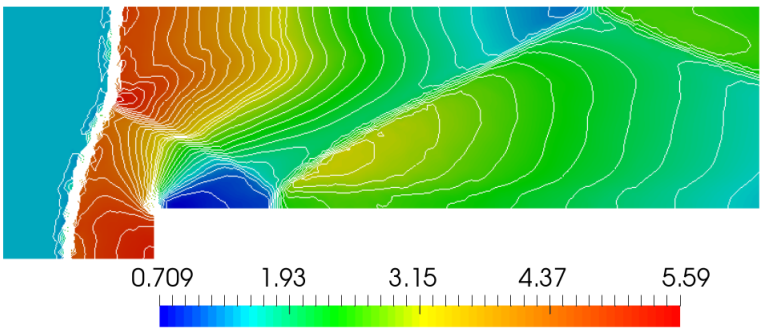
(b) CAU



(c) $SUPG + YZ\beta$



(d) $NMV1$



(e) $NMV2$

Figure 3.23 – Wind tunnel problem: density distribution 2D solution at time $t = 3.0$.

The computational performance of the methods is shown in Table 3.7. The nonlinear process goes on until the convergence criteria is satisfied or the maximum number of iteration reaches 10. We can see in the fourth column that all methods converge on average with fewer than 10 iterations, the only exception is the CAU method that spend approximately 10 iterations in each linearization process. The NMV1 and NMV2 methods need fewer GMRES iterations and less CPU time than the CAU and SUPG + YZβ methods. In addition, the NMV1 (NMV2) method requires approximately 58%(68%) of the CPU time required by the CAU method, and approximately 72%(85%) of the CPU time required by the SUPG + YZβ method.

Table 3.7 – Computational performance - Wind tunnel with nonlinear tolerance of 10^{-2}.

Methods	GMRES Iter.	Nonlinear Iter. (NL)	NL/3,000	CPU Time (s)
CAU	565,776	29,982	9.994	719.95631
SUPG+YZβ	348,526	16,874	5.625	629.05130
NMV1	140,064	18,712	6.237	466.58204
NMV2	212,027	21,843	7.281	602.53053

Figure 3.24 shows the time evolution of the density residual $L^2(\Omega)$-norm. The residual sequence obtained by the NMV1 and NMV2 methods have a similar behavior remaining approximately constant at the order of 10^{-4}, the residual sequence of the CAU remains approximately constant at the order of 10^2, whereas the SUPG + YZβ one oscillates near 10^{-1} during the time evolution process.

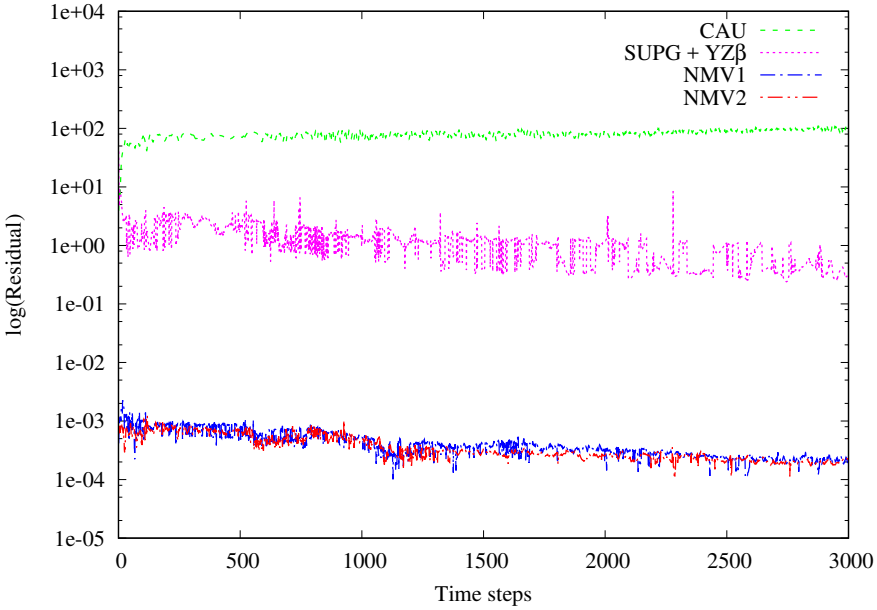


Figure 3.24 – Wind tunnel problem: Residual of density.

4 Time integration scheme

In this chapter we introduce a different time integration scheme to solve the resulting system of ordinary differential equation (or system of differential algebraic equations (DAE)) originated from the spatial stabilization strategies addressed in Chapter 3. It is well known that the system of Euler equations is stiff, a concept indicating that different physical phenomena acting on different time scales occur simultaneously, difficulting its numerical solution. This problem can be solved by using methods that have the stiff decay property. For this purpose, we propose an unusual predictor-corrector method based on Backward Differentiation Formulas (BDF), where the prediction phase is based on an extrapolation process.

4.1 Stiff Initial-Value Problems

Stiff differential ordinary equations are known since the early 1950s (see Curtiss and Hirschfelder (1952)). Those equations presented a great challenge to numerical methods in that time. Since then a great effort has been made by the scientific community to analyze this type of problem, and as a consequence, many numerical methods have been proposed to solve stiff problems. Curtiss and Hirschfelder (1952) observed that explicit methods are not adequate to solve ordinary differential equations that model certain chemical reactions. It was introduced the notation *stiffness* to indicate chemical reactions in which the fastly reaction components reach their equilibrium in a very short time and the slowly changing components are more or less fixed, i.e., stiff.

Even though stiffness is phenomenologically well understood, according to Cash (2003),

> *One of the major difficulties associated with the study of stiff differential system is that a good mathematical definition of the concept of stiffness does not exist.*

According to Hairer and Wanner (1996), (as cited in Söderlind et al. (2015)),

> *While the intuitive meaning of stiff is clear to all specialists, much controversy is going on about its correct mathematical definition [...]. The most pragmatical opinion is historically the first one (Curtissand Hirschfelder 1952): stiff equations are equations where certain implicit methods, in particular BDF, perform better, usually tremendously better, than explicit ones.*

There is no unique definition of stiffness in the literature. More recently, a new concept of stiffness in given by Söderlind et al. (2015). Stiff equations are multiscale problems, since they represent coupled physical system having components which vary on very different time-scales (Cash, 2003). The stiffness of a system of differential algebraic equations (DAE) is related to the decay rate of their solution, that is, due to large disparity in the timescales (Knoll and Keyes, 2004). A DAE system has different decay rates for each state variable, and are related to the eigenvalues of the Jacobian matrix of the system.

An important strategy for solving stiff problems is the application of time integration methods with a decay stiff property. A class of methods based on backward differences, for stiff problems, was discovered by Curtiss and Hirschfelder (1952), and their importance for stiff problems has been recognized from the work of Gear (1971). These methods belong to a class of implicit multistep time integrators known as the Backward Differentiation Formulas (BDF). The BDF methods of order k with fixed time step are defined by

$$\sum_{j=1}^{k} \tfrac{1}{j} \nabla^j y_{n+1} = \Delta t f_{n+1}, \tag{4.1}$$

to solve the problem $y' = f(t, y)$ in (t_0, t_f), where y_{n+1} is the approximation of $y(t_{n+1})$, $f_{n+1} = f(t_{n+1}, y_{n+1})$, and Δt is the time-step. For $k = 1$ and $k = 2$, we obtain, respectively:

$$\text{BDF-1} \quad : \quad y_{n+1} - y_n = \Delta t f_{n+1}; \tag{4.2}$$

$$\text{BDF-2} \quad : \quad \tfrac{3}{2} y_{n+1} - 2y_n + \tfrac{1}{2} y_{n-1} = \Delta t f_{n+1}. \tag{4.3}$$

The BDF-1 is the first-order implicit backward Euler method while BDF-2 is the second-order BDF method, or the Gear time-stepping scheme.

The stability of time integration methods in the classical sense is one in which the stability domain (Definition 1, see below) contains the complex negative half-plane. This kind of stability is called *A-stability*. The Crank-Nicolson or trapezoidal rule methods (implicit Adams method of order 2) are A-stables, for example. This property guarantees dissipation of numerical errors with time, but it gives no indication as to the rate at which this dissipation occurs (Hay et al., 2015).

In order to describe the stability domain, it is necessary to define the stability function of the method. This is done applying the BDF-1 method (Eq. (4.2)) in the scalar equation

$$y' = f(t, y) = \lambda y,$$

with $\lambda \in \mathbb{C}$, known as *Dahlquist test equation* (or *linear test equation*),

$$\begin{aligned} y_{n+1} &= y_n + \Delta t f_{n+1}, \\ &= y_n + \Delta t \lambda y_{n+1}. \end{aligned}$$

The last equation results in

$$\begin{aligned} y_{n+1} &= y_n \left(\frac{1}{1 - \Delta t \lambda} \right) \\ &= y_0 \left(\frac{1}{1 - \Delta t \lambda} \right)^{n+1}. \end{aligned}$$

Setting $z = \Delta t \lambda$ and $y_0 = 1$, we arrive at

$$y_{n+1} = R(z)^{n+1}, \tag{4.4}$$

with $R(z) = \left(\frac{1}{1 - z} \right)$.

Definition 1 (Hairer et al., 2010) *The function $R(z)$, defined in (4.4), is called the stability function. The set*

$$S = \{ z \in \mathbb{C}; \quad |R(z)| \leqslant 1 \}$$

is called the stability domain of the method.

The BDF-1 and BDF-2 methods have a decay stiff property and are extremely fast at dissipating numerical errors when applied within their stability regions (Hay et al., 2015). This property is called *L−stability* and is defined as follow,

Definition 2 (Hairer et al., 2010) *A method is called L−stable if it is A−stable and if in addition*

$$\lim_{z \to \infty} R(z) = 0.$$

In the next section is presented a predictor-corrector scheme for solving DAE system originated from the NMV1 and NMV2 methods (described in Chapter 3), based on the BDF-2 method.

4.2 A predictor-corrector scheme based on the BDF-2

A predictor–corrector method for solving ordinary differential equations typically uses an explicit method for the predictor step and an implicit method for the corrector step, both of the same accurate order (Hairer and Wanner, 2010). According to Hairer et al. (2010), if the implicit equations are solved by the predictor-corrector scheme, a good deal of the stability is lost and the *A*-stability of the implicit method can be destroyed. For example, denoting by S_{exp}, $S_{pred-cor}$, S_{imp} the stability domains of the explicit, predictor-corrector and implicit approaches, respectively, we will have the following relation

$$S_{exp} \subsetneq S_{pred-cor} \subsetneq S_{imp}, \tag{4.5}$$

that is, the stability domain of the predictor-corrector scheme is a subset of the stability domain of the implicit method used in the corrector phase. According to Hay et al. (2015), a good way for solving implicit BDF equations with predictor-corrector approach, consists of, from the polynomial interpolation of the solution at previous time instants, extrapolating the solution (predictor) at the next time instant. Then, this solution is replaced in the implicit BDF method, obtaining a corrected solution. Using this methodology, we propose a new predictor-corrector scheme based on BDF-2 for the NMV1 and NMV2 methods.

Consider the DAE system (3.26)-(3.27) rewritten as follow,

$$\begin{aligned} \boldsymbol{M}_{hh}\dot{\boldsymbol{U}}_h^{n+1} + \boldsymbol{M}_{hb}\dot{\boldsymbol{U}}_b^{n+1} + \boldsymbol{K}_{hh}\boldsymbol{U}_h^{n+1} + \boldsymbol{K}_{hb}\boldsymbol{U}_b^{n+1} &= \boldsymbol{0}_h, \quad (4.6)\\ \boldsymbol{M}_{bh}\dot{\boldsymbol{U}}_h^{n+1} + \boldsymbol{M}_{bb}\dot{\boldsymbol{U}}_b^{n+1} + \boldsymbol{K}_{bh}\boldsymbol{U}_h^{n+1} + \boldsymbol{K}_{bb}\boldsymbol{U}_b^{n+1} &= \boldsymbol{0}_b. \quad (4.7) \end{aligned}$$

By plugging the expression of the BDF-2 (Eq. (4.3)), for $\dot{U}_b^{n+1}$, into Eq. (4.6) and (4.7), we obtain

$$\begin{aligned} \boldsymbol{M}_{hh}\dot{U}_h^{n+1} + \boldsymbol{M}_{hb}\left(\frac{3U_b^{n+1} - 4U_b^n + U_b^{n-1}}{2\Delta t}\right) + \boldsymbol{K}_{hh}U_h^{n+1} + \boldsymbol{K}_{hb}U_b^{n+1} &= 0_h,\\ \boldsymbol{M}_{bh}\dot{U}_h^{n+1} + \boldsymbol{M}_{bb}\left(\frac{3U_b^{n+1} - 4U_b^n + U_b^{n-1}}{2\Delta t}\right) + \boldsymbol{K}_{bh}U_h^{n+1} + \boldsymbol{K}_{bb}U_b^{n+1} &= 0_b. \end{aligned}$$

Arranging the terms yield

$$\begin{aligned} \tfrac{2}{3}\Delta t\boldsymbol{M}_{hh}\dot{U}_h^{n+1} + \boldsymbol{N}_1 U_b^{n+1} + \tfrac{2}{3}\Delta t\boldsymbol{K}_{hh}U_h^{n+1} &= \tfrac{2}{3}\Delta t F_h^{n+1} + \boldsymbol{M}_{hb}\left(\tfrac{4}{3}U_b^n - \tfrac{1}{3}U_b^{n-1}\right), \quad (4.8)\\ \tfrac{2}{3}\Delta t\boldsymbol{M}_{bh}\dot{U}_h^{n+1} + \boldsymbol{N}_2 U_b^{n+1} + \tfrac{2}{3}\Delta t\boldsymbol{K}_{bh}U_h^{n+1} &= \tfrac{2}{3}\Delta t F_b^{n+1} + \boldsymbol{M}_{bb}\left(\tfrac{4}{3}U_b^n - \tfrac{1}{3}U_b^{n-1}\right), \quad (4.9) \end{aligned}$$

where
$\boldsymbol{N}_1 = \boldsymbol{M}_{hb} + \frac{2}{3}\Delta t\boldsymbol{K}_{hb}$,
$\boldsymbol{N}_2 = \boldsymbol{M}_{bb} + \frac{2}{3}\Delta t\boldsymbol{K}_{bb}$.
Isolating the unknown of the micro scale, Eq. (4.9), at time $n+1$, we obtain an expression for the micro scale solution,

$$U_b^{n+1} = \boldsymbol{N}_2^{-1}\left[\boldsymbol{M}_{bb}\left(\tfrac{4}{3}U_b^n - \tfrac{1}{3}U_b^{n-1}\right) - \tfrac{2}{3}\Delta t\left(\boldsymbol{M}_{bh}\dot{U}_h^{n+1} + \boldsymbol{K}_{bh}U_h^{n+1}\right)\right]. \quad (4.10)$$

Replacing Eq. (4.10) into Eq. (4.8), it results in the system of DAE for resolved scale

$$\boldsymbol{M}\dot{U}_h^{n+1} + \boldsymbol{K}U_h^{n+1} = \boldsymbol{N}\left(\tfrac{4}{3}U_b^n - \tfrac{1}{3}U_b^{n-1}\right), \quad (4.11)$$

where

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{M}_{hh} - \boldsymbol{N}_1\boldsymbol{N}_2^{-1}\boldsymbol{M}_{bh}; \quad (4.12)\\ \boldsymbol{K} &= \boldsymbol{K}_{hh} - \boldsymbol{N}_1\boldsymbol{N}_2^{-1}\boldsymbol{K}_{bh}; \quad (4.13)\\ \boldsymbol{N} &= \tfrac{3}{2\Delta t}\left(\boldsymbol{M}_{hb} - \boldsymbol{N}_1\boldsymbol{N}_2^{-1}\boldsymbol{M}_{bb}\right). \quad (4.14) \end{aligned}$$

The Eq. (4.11) can be re-written in as follows:

$$\boldsymbol{M}\dot{U}_h^{n+1} = \boldsymbol{F}(U_h^{n+1}, U_b^n, U_b^{n-1}, t^{n+1}), \tag{4.15}$$

with

$$\boldsymbol{F}(U_h^{n+1}, U_b^n, U_b^{n-1}, t^{n+1}) = -\boldsymbol{K}U_h^{n+1} + \boldsymbol{N}\left(\tfrac{4}{3}U_b^n - \tfrac{1}{3}U_b^{n-1}\right).$$

Equation (4.15) describes a implicit DAE system for the resolved scale, where the right-hand side depends on U_b^n, that can be avaluated using (4.10). Applying BDF-2 expression (Eq. (4.3)) into (Eq. (4.15)) we obtain the following nonlinear resolved scale problem

$$\boldsymbol{R}_h(U_h^{n+1}) = \boldsymbol{0},$$

where the resolved BDF-2 operator $\boldsymbol{R}_h$ is given by

$$\boldsymbol{R}_h(U_h^{n+1}) = \boldsymbol{M}\left(\tfrac{4}{3}U_h^n - \tfrac{1}{3}U_h^{n-1}\right) + \tfrac{2}{3}\Delta t\boldsymbol{F}(U_h^{n+1}, U_b^n, U_b^{n-1}, t^{n+1}) - \boldsymbol{M}U_h^{n+1}, \tag{4.16}$$

whose Jacobian matrix is

$$\boldsymbol{J}_h = \frac{\partial \boldsymbol{R}_h}{\partial U_h^{n+1}} = -\left(\boldsymbol{M} + \tfrac{2}{3}\Delta t\boldsymbol{K}\right). \tag{4.17}$$

Describing a Newton method to solve the nonlinear process at time $n+1$, we obtain

$$\boldsymbol{J}_h\Delta U_h^{n+1,i+1} = -\boldsymbol{R}_h(U_h^{n+1,i}), \tag{4.18}$$

or

$$\left(\boldsymbol{M} + \tfrac{2}{3}\Delta t\boldsymbol{K}\right)\Delta U_h^{n+1,i+1} = \boldsymbol{M}\left(\tfrac{4}{3}U_h^n - \tfrac{1}{3}U_h^{n-1}\right) + \tfrac{2}{3}\Delta t\boldsymbol{F} - \boldsymbol{M}U_h^{n+1,i}. \tag{4.19}$$

The last equation is defined locally on each element. After the assembly process it generates a global system associated. The global system can be solved using (4.10) to calculate the vector $\boldsymbol{F}$, but we do not use this strategy here. The same process followed for the resolved scales is used in the unresolved ones to obtain a similar problem that will be solved exactly on each element.

From Eq. (4.7), we obtain the DAE system for the unresolved scale,

$$\boldsymbol{M}_{bb}\dot{U}_b^{n+1} = -\left(\boldsymbol{M}_{bh}\dot{U}_h^{n+1} + \boldsymbol{K}_{bh}U_h^{n+1} + \boldsymbol{K}_{bb}U_b^{n+1}\right). \tag{4.20}$$

Applying the BDF-2 expression (Eq. (4.3)) in (4.20), we obtain the BDF-2 operator for unresolved scale

$$\begin{aligned}\boldsymbol{R}_b(U_b^{n+1,i}) \;=\; & \boldsymbol{M}_{bb}\left(\tfrac{4}{3}U_b^n - \tfrac{1}{3}U_b^{n-1}\right) - \tfrac{2}{3}\Delta t\left(\boldsymbol{M}_{bh}\dot{U}_h^{n+1,i+1} + \boldsymbol{K}_{bh}U_h^{n+1,i+1} + \boldsymbol{K}_{bb}U_b^{n+1,i}\right) \\ & -\boldsymbol{M}_{bb}U_b^{n+1,i}.\end{aligned} \tag{4.21}$$

Analogously to the macro scale, by Newton's method, we arrive at

$$\boldsymbol{J}_b\Delta U_b^{n+1,i+1} = -\boldsymbol{R}_b(U_b^{n+1,i}), \tag{4.22}$$

or

$$\left(\boldsymbol{M}_{bb} + \tfrac{2}{3}\Delta t \boldsymbol{K}_{bb}\right) \Delta U_h^{n+1,i+1} = \boldsymbol{M}_{bb}\left(\tfrac{4}{3}U_b^n - \tfrac{1}{3}U_b^{n-1}\right) - \tfrac{2}{3}\Delta t\left(\boldsymbol{M}_{bh}\dot{U}_h^{n+1,i+1} + \boldsymbol{K}_{bh}U_h^{n+1,i+1}\right) - \left(\boldsymbol{M}_{bb} + \tfrac{2}{3}\Delta t \boldsymbol{K}_{bb}\right) U_b^{n+1,i}. \tag{4.23}$$

The global system associated to (4.19) and the local system (4.23) represent the problems to be solved in each nonlinear iterative process along of the time, for the resolved and unresolved scales, respectively. This implies that the marching in time occurs simultaneously in both scales. For the resolved scale problem (4.19), in each nonlinear iteration the global linear system is solved by the GMRES method, whereas the linear system on the unresolved scale, described by (4.23), is solved exactly on each element (locally), because we need only to invert the matrix of order 4,

$$\boldsymbol{M}_{bb} + \tfrac{2}{3}\Delta t \boldsymbol{K}_{bb} = \left(M_{bb} + \tfrac{2}{3}\Delta t K_{bb}\right) \boldsymbol{I}_4, \tag{4.24}$$

where $M_{bb} + \frac{2}{3}\Delta t K_{bb}$ is a nonzero real number (for details see Eqs (3.20) and (3.25)), in the Eq (4.23).

To obtain a solution in the prediction phase, we consider the Newton interpolation polynomial q of degree 2 interpolating U_h^{n-2}, U_h^{n-1}, and U_h^n at time instants t_{n-2}, t_{n-1}, and t_n, respectively. We can write the polynomial q as follows (Quarteroni and Saleri, 2006),

$$q(t) = U_h^{n-2} + \triangle U_h^{n-2}(t - t_{n-2}) + \triangle^2 U_h^{n-2}(t - t_{n-2})(t - t_{n-1}), \tag{4.25}$$

where the symbol $\triangle$ is the Newton's divided difference operator. The same prediction is applied on the micro scale.

According to the explanation in this section we define the Algorithm 2, a predictor-corrector scheme based on BDF-2 for our multiscale approaches. In Algorithm 2, the lines 6 and 7 shows the prediction phase through Newton's polynomial extrapolation for macro and micro scales, in lines 9-18 we have the correction phase based on BDF-2, where the solution is corrected by the BDF-2 operator (line 10). This algorithm can be initialized with a classical predictor-corrector scheme based on BDF-1.

4.3 Convergence rates study

As discussed before, the methods in the predictor-corrector form influences the size of stability domain (Eq. (4.5)). Furthermore, the predictor-corrector method has generally the stability of the predictor method and accuracy order of the corrector method (Quarteroni and Saleri, 2006; Quarteroni et al., 2006), i.e., the classical application in the predictor-corrector form causes it to lose stability properties. In this section we evaluate the convergence rates of the Algorithm 1, described in Section 3.5, and the Algorithm 2, described in Section 4.2, both predictor-corrector schemes.

Algorithm 2 Predictor-Corrector algorithm based on the BDF2 (PC-BDF2)

input: U_h^0, U_h^1, U_h^2, U_b^0, U_b^1, and U_b^2
$t \leftarrow 0$
$n \leftarrow 0$
repeat
  $t \leftarrow t + \Delta t$
  $U_h^{n+1,0} \leftarrow U_h^{n-2} + 3\mathbb{\Delta} U_h^{n-2}\Delta t + 6\mathbb{\Delta}^2 U_h^{n-2}\Delta t^2$
  $U_b^{n+1,0} \leftarrow U_b^{n-2} + 3\mathbb{\Delta} U_b^{n-2}\Delta t + 6\mathbb{\Delta}^2 U_b^{n-2}\Delta t^2$
  $i \leftarrow 0$
  repeat
    $\boldsymbol{R}_h^{n+1,i} \leftarrow \boldsymbol{M}\left(\frac{4}{3}U_h^n - \frac{1}{3}U_h^{n-1}\right) + \frac{2}{3}\Delta t\left[-\boldsymbol{K}U_h^{n+1} + \boldsymbol{N}\left(\frac{4}{3}U_b^n - \frac{1}{3}U_b^{n-1}\right)\right] - \boldsymbol{M}U_h^{n+1,i}$ ▷ BDF2 operator
    $\boldsymbol{J}_h \Delta U_h^{n+1,i+1} = \boldsymbol{R}_h^{n+1,i}$
    $U_h^{n+1,i+1} \leftarrow U_h^{n+1,i} + \Delta U_h^{n+1,i+1}$ ▷ update macro solution
    $\dot{U}_h^{n+1,i+1} \leftarrow \frac{1}{\Delta t}\left(\frac{3}{2}U_h^{n+1,i+1} - 2U_h^n + \frac{1}{2}U_h^{n-1}\right)$
    $\boldsymbol{R}_b^{n+1,i} \leftarrow \boldsymbol{M}_{bb}\left(\frac{4}{3}U_b^n - \frac{1}{3}U_b^{n-1}\right) - \frac{2}{3}\Delta t\left(\boldsymbol{M}_{bh}\dot{U}_h^{n+1,i+1} + \boldsymbol{K}_{bh}U_h^{n+1,i+1}\right) - \boldsymbol{N}_2 U_b^{n+1,i}$
    $\Delta U_b^{n+1,i+1} \leftarrow \boldsymbol{N}_2^{-1}\boldsymbol{R}_b^{n+1,i}$
    $U_b^{n+1,i+1} \leftarrow U_b^{n+1,i} + \Delta U_b^{n+1,i+1}$ ▷ update micro solution
    $i \leftarrow i + 1$
  until $\frac{||U_h^{n+1,i+1} - U_h^{n+1,i}||_2}{||U_h^{n+1,i+1}||_2} < tol_i$ **or** $i > i_{\text{MAX}}$
  $n \leftarrow n + 1$
until $t < t_f$ ▷ t_f: final time

The Algorithm 1 defines a predictor-corrector based on the one-step α-method. The α-method, described by

$$U_h^{n+1} = U_h^n + (1-\alpha)\Delta t\dot{U}_h^n + \alpha\Delta t\Delta\dot{U}_h^{n+1}, \tag{4.26}$$

is a second order accurate scheme just for $\alpha = 1/2$ and first order accurate for any other values of α (Förster, 2007). The Algorithm 1 is not defined in the form: explicit prediction and implicit correction. Instead, the $\alpha-$method (Eq. (4.26)) is truncated in its implicit part for the prediction $(U_h^n + (1-\alpha)\Delta t\dot{U}_h^n)$. On the other hand, the correction is done by adding an increment $(\alpha\Delta t\Delta\dot{U}_h^{n+1})$. We are going to see, through a convergence study applying Algorithm 1, that the second order ($\alpha = 1/2$) accuracy is lost in this way of prediction-correction.

We evaluate the temporal convergence rates for the Algorithm 1 and Algorithm 2, using a parabolic problem with smooth exact solution: find $u(x, y, t)$ such that

$$\begin{cases} \dfrac{\partial u}{\partial t} - \varepsilon\Delta u = 0, \text{ in } \Omega \times (0, t_f]; \\ u(x, y, t) = 0, \text{ on } \partial\Omega; \\ u(x, y, 0) = \sin(\frac{\pi x}{2})\sin(\frac{\pi y}{2}), \end{cases} \tag{4.27}$$

whose solution is given by

$$u(x, y, t) = \exp(\varepsilon \frac{\pi^2}{4} t) \sin(\frac{\pi x}{2}) \sin(\frac{\pi y}{2}). \tag{4.28}$$

We can verify the function u (Eq. (4.28)) is smooth enough to estimate the convergence rate. In our tests we have set $\varepsilon = 0.1$.

The spatial computational domain Ω is the square with $0 < x < 2$ and $0 < y < 2$. The test were carried out using a triangular uniform mesh with 1,002,001 nodes and 2,000,000 elements, whose element length is $h = 0.002$. The very refined mesh is important so that the approximation in space does not influence the approximation in time. The time step size is taken as $\Delta t = 1.0;\ 0.5;\ 0.25;\ 0.125$.

Figure 4.1 shows the estimation of the convergence rate, in the $L^2(\Omega)$-norm, of the time integration methods, PC-std (Algorithm 1 for $\alpha = 1/2$) and PC-BDF2 (Algorithm 2). The $L^2(\Omega)$-norm error (Problem (4.27)) is calculated at the final time $t_f = 5.0$. We can verify that the prediction-correction form presented in Algorithm 1 destroys the second order of accuracy coming from of the $\alpha-$method, for $\alpha = 1/2$. On the other hand, for PC-BDF2 method, we expected a convergence rate of approximately 2, but we found a higher value as shown in Fig. 4.1, characterizing a hyper-convergence effect. A numerical analysis must be done to clearly understand this phenomenon.

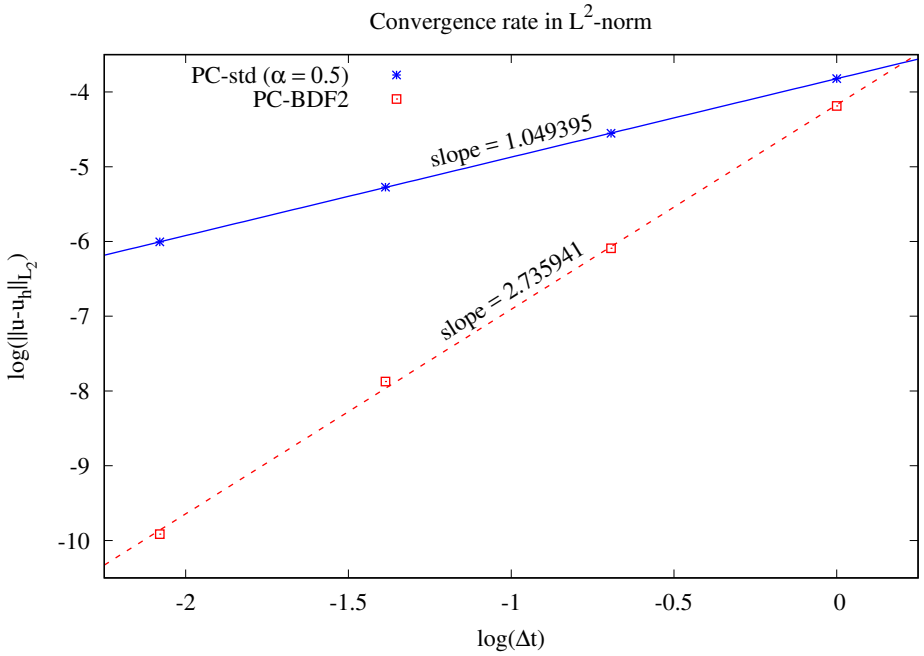


Figure 4.1 – Time integration schemes: $L^2(\Omega)$-norm convergence rates at $t_f = 5.0$.

4.4 Numerical results

In order to evaluate the efficiency of the predictor-corrector based on BDF2 proposed in Algorithm 2, we present the numerical experiments considering two transonic problems: "Sod's shock tube" and "explosion". We compare the NMV methods combined with the time integration given by Algorithm 1 – named here standard predictor-corrector (PC-std)–, and Algorithm 2 – predictor-corrector based on BDF2 (PC-BDF2). In all experiments in this section, we set up the GMRES solver with 30 vectors to restart, tolerance is equal to 10^{-5}, the time-step size is 10^{-3}, and tol_δ in Eq. (3.13) is set equal to 10^{-8}. The tests were performed on a dedicated machine as defined in Section 3.6.

4.4.1 Shock tube problem

We simulate again the shock tube problem as described in Subsection 3.6.1 with nonlinear tolerance of 10^{-3} and the maximum number of iterations equals 3, in order to evaluate the PC-BDF2 because the stop criteria by norm (Algorithm 2, line 18) is satisfied more quickly. In this experiment, the NMV methods combined with the PC-BDF2 scheme generally need just one nonlinear iteration to satisfy the tolerance, whereas the NMV methods combined with the PC-std do not satisfy nonlinear tolerance spending the maximum number of iterations.

Table 4.1 shows the computational performance for the NMV methods combined with Algorithm 1 (PC-std) and Algorithm 2 (PC-BDF2). We can see, with nonlinear correction tolerance 10^{-3} and maximum number of iterations equal to 3, the NMV methods combined with PC-BDF2 spend half of CPU time and GMRES iterations than the PC-std, i.e., the PC-BDF2 improves the linearization procedure with the extrapolation process that yields a good initial guess for the Newton iterative process at each time step (Hay et al., 2015). The PC-BDF2 needs fewer corrections to obtain a solution slightly more accurate than that with the PC-std, as we can see in Fig. 4.2-4.3 and through in the L_2-norm error (Table 4.2). Furthermore, the L_2-norm of the density residual obtained with the PC-BDF2 has the same behavior of that with the PC-std, but with smaller values (Fig. 4.4-4.5).

4.4.2 Explosion problem

The explosion problem, as described in Subsection 3.6.4 with nonlinear tolerance of 10^{-3} and the maximum number of iterations equals 3, is simulated again here to evaluate the NMV methods combined with the PC-BDF2. As in shock tube problem (Subsection 4.4.1), the NMV methods combined with the PC-BDF2 scheme generally need just one nonlinear iteration to satisfy the nonlinear tolerance, whereas the NMV methods combined with the PC-std do not satisfy nonlinear tolerance spending the maximum number of iterations.

Table 4.1 – Computational performance - Shock tube with nonlinear tolerance of 10^{-3} and the maximum number of iterations is equal to 3.

Methods	GMRES Iterations	CPU Time (s)
NMV1(PC-BDF2)	1,767	0.200265
NMV1(PC-std)	3,909	0.395294
NMV2(PC-BDF2)	1,948	0.217530
NMV2(PC-std)	4,610	0.463788

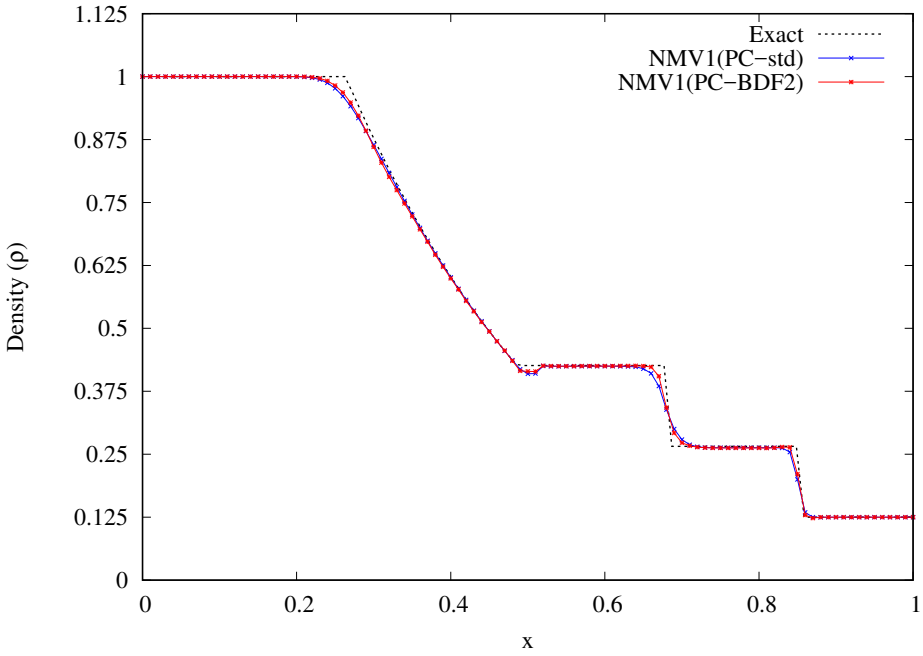


Figure 4.2 – Shock tube problem with the NMV1 method: Density profile along $y = 0.01$. Iterative procedure: $tol_i = 10^{-3}$ and $i_{\max} = 3$.

The PC-BDF2 spends almost one third of the CPU time and GMRES iterations than the PC-std (Table 4.3), i.e., in general, the PC-BDF2 needs only one correction step to obtain a solution slightly more accurate (Fig. 4.6-4.7). Again, we can see that the strategy of extrapolation used in Algorithm 2 yields a good initial guess for the Newton's method at each time step. Figures 4.8 and 4.9 shows the convergence history in L_2-norm of the density residual of the explosion problem. After 50 steps, the sequence of the all methods remainder approximately constant, but for the solutions with the PC-BDF2 scheme they present smaller values.

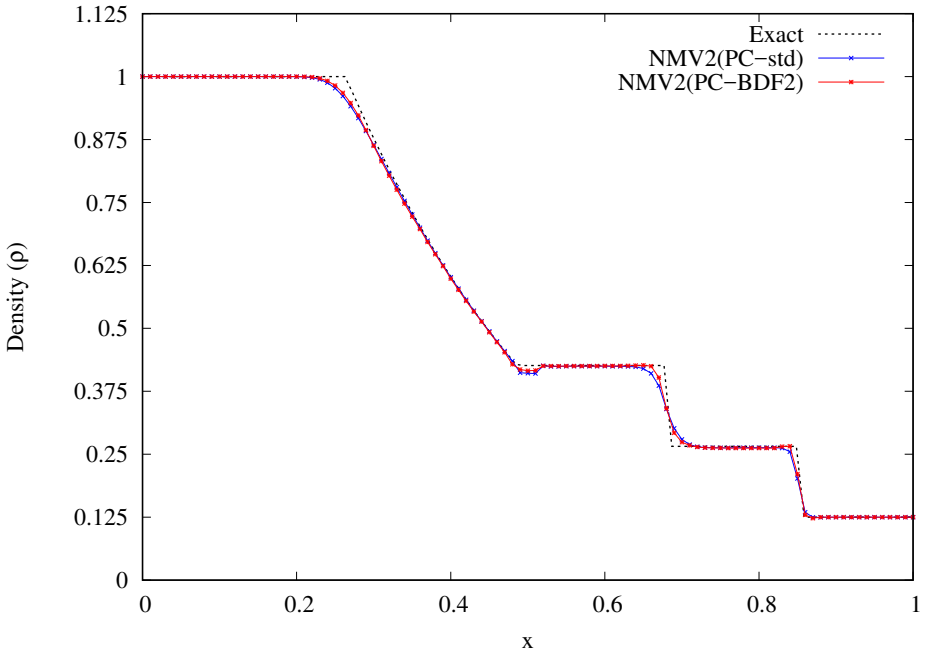


Figure 4.3 – Shock tube problem with the NMV2 method: Density profile along $y = 0.01$. Iterative procedure: $tol_i = 10^{-3}$ and $i_{\max} = 3$.

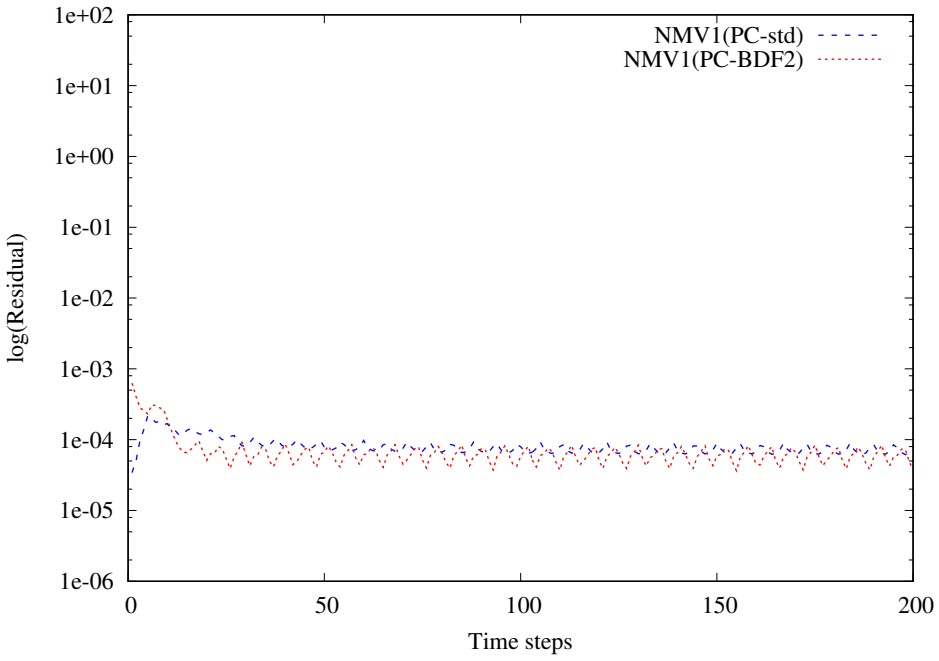


Figure 4.4 – Shock tube problem with the NMV1 method: Residual of density. Iterative procedure: $tol_i = 10^{-3}$ and $i_{\max} = 3$.

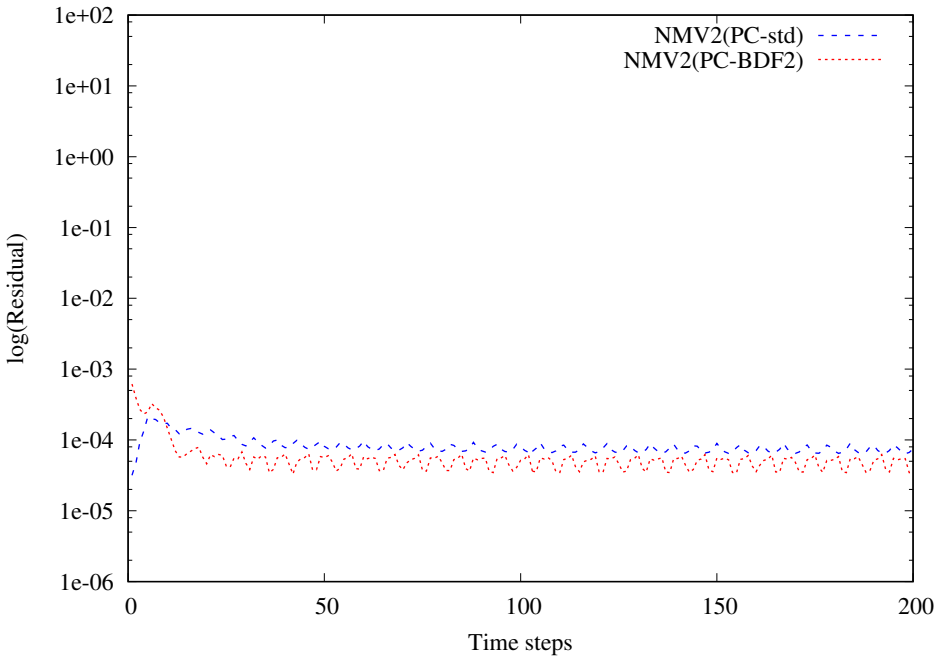


Figure 4.5 – Shock tube problem with the NMV2 method: Residual of density. Iterative procedure: $tol_i = 10^{-3}$ and $i_{\max} = 3$.

Table 4.2 – Shock tube: L^2 norm error.

Methods	$\|\|\rho - \rho_h\|\|_{L_2}$
NMV1(PC-BDF2)	1.625648E-02
NMV1(PC-std)	1.683335E-02
NMV2(PC-BDF2)	1.641194E-02
NMV2(PC-std)	1.690536E-02

Table 4.3 – Computational performance - Explosion problem with nonlinear tolerance of 10^{-3} and maximum number of iterations is equal to 3.

Methods	GMRES Iterations	CPU Time (s)
NMV1(PC-BDF2)	2,176	21.687054
NMV1(PC-std)	5,685	54.844225
NMV2(PC-BDF2)	2,457	23.860522
NMV2(PC-std)	5,871	59.717147

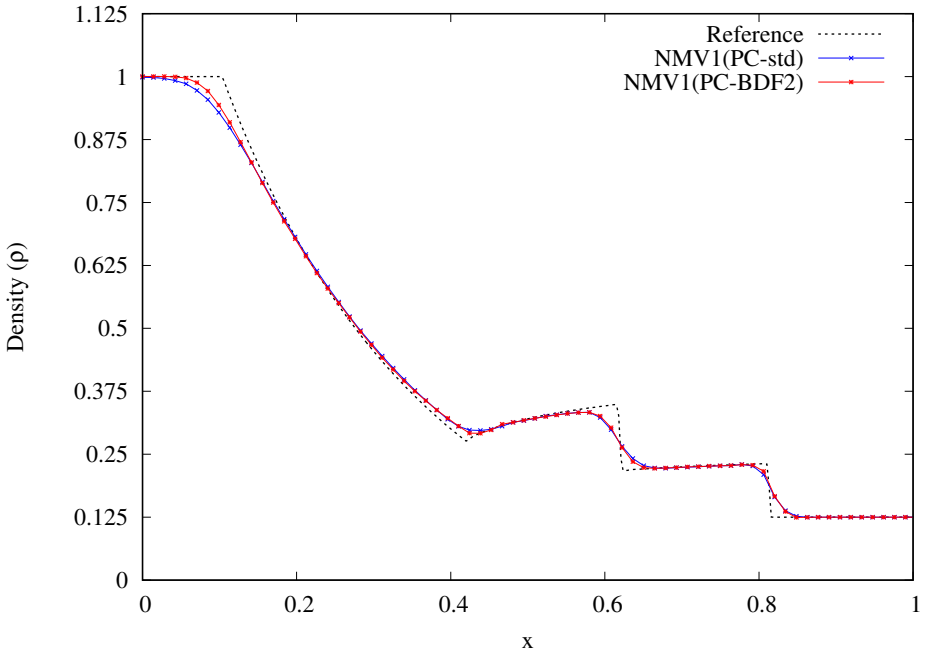


Figure 4.6 – Explosion problem with the NMV1 method: Comparisons of radial variations of density. Iterative procedure: $tol_i = 10^{-3}$ and $i_{\max} = 3$.

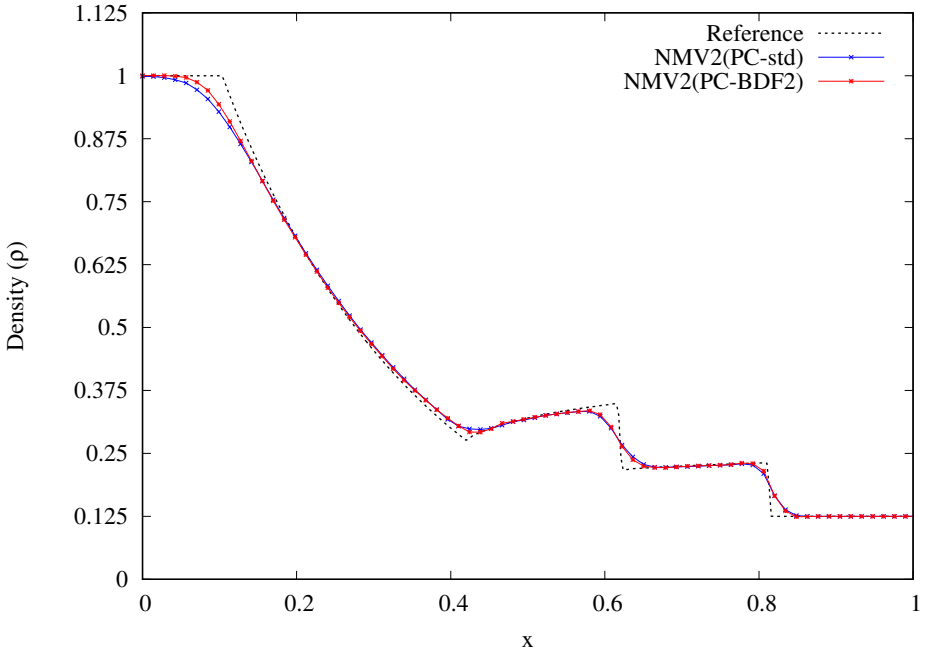


Figure 4.7 – Explosion problem with the NMV2 method: Comparisons of radial variations of density. Iterative procedure: $tol_i = 10^{-3}$ and $i_{\max} = 3$.

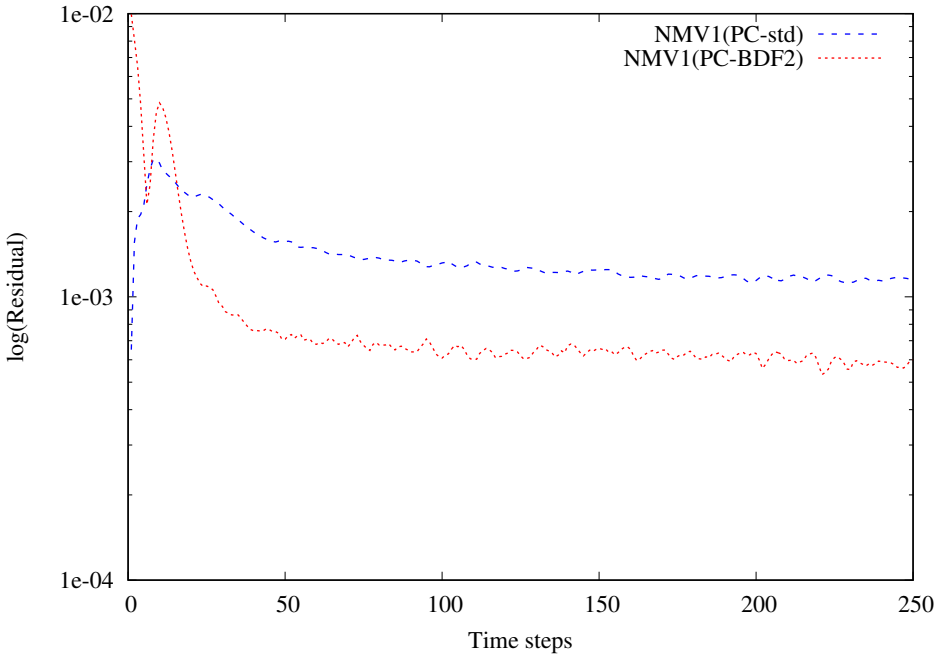


Figure 4.8 – Explosion problem with the NMV1 method: Residual of density. Iterative procedure: $tol_i = 10^{-3}$ and $i_{\max} = 3$.

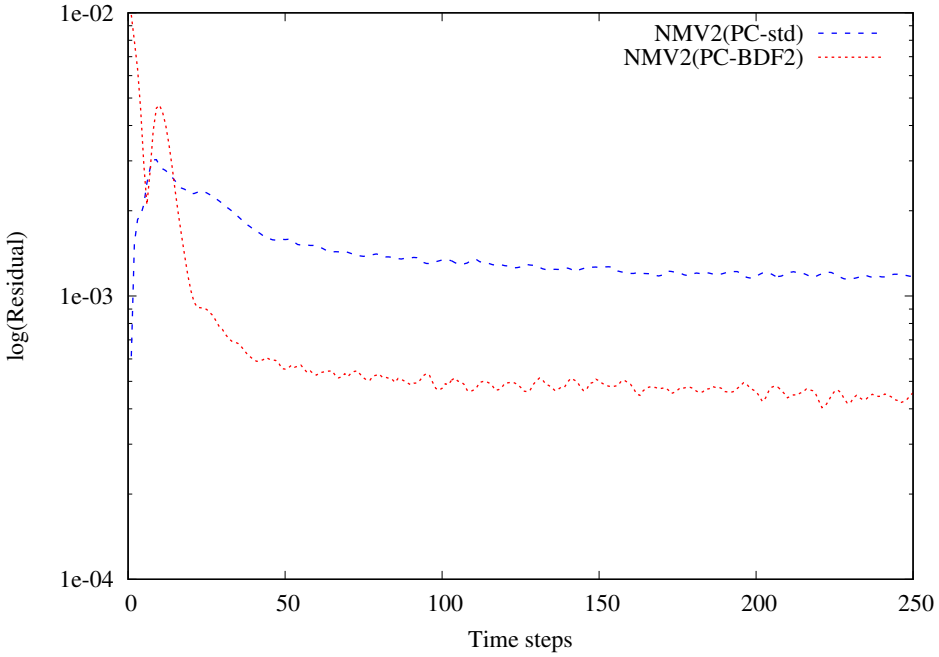


Figure 4.9 – Explosion problem with the NMV2 method: Residual of density. Iterative procedure: $tol_i = 10^{-3}$ and $i_{\max} = 3$.

5 Local preconditioning and stiffness of the Euler equations

In this chapter we introduce a preconditioned variational multiscale method applied to the Euler equations. Two local preconditioning techniques are combined with the NMV methods in order to deal with the stiffness property of compressible flow. We focus only on steady state problems. A numerical experiment was conducted on a NACA 0012 airfoil in order to examine the stability of the method in terms of the Mach number variations, specially in low Mach number limit.

5.1 Stiffness of the Euler equations

There are many challenges in developing numerical methods for solving problems from low to high speed compressible flows. Numerical methods addressed for solving low speed are usually pressure-based, since the flow is approaching to the incompressibility. In transonic and supersonic regimes the numerical methods generally are density-based. Those last schemes suffer with undesirable effects of low speed flow including low convergence speed and loss of accuracy (Li and Xiang, 2013). This phenomenon occurs because the fast waves impose their small time steps to the slow ones (Ginard et al., 2016b).

As discussed in Chapter 4, a system of DAE is stiff due to the large disparity in their timescales (Knoll and Keyes, 2004). In the same way, according to Bassi et al. (2009), the system of the compressible Euler equations is also stiff if it covers a wide range of timescales. The stiffness of DAE systems is measured through the eigenvalues ratio of the Jacobian related with the problem (Ashino et al., 2000; Hairer et al., 2010). In the context of conservation laws, precisely, compressible Euler equations, the stiffness is measured through of the disparities related to the characteristic propagation speeds of the system, that are given by the eigenvalues of the Euler flux Jacobian (Lopez et al., 2012; Ginard et al., 2016b). In both situations (DAE system and Euler equations) the term stiff defines the same behavior: disparities of time scale.

In the Euler equations context, another term used to refer to stiffness is ill-conditioning (Ginard et al., 2016b). In this case, a system of Euler equations is stiff or ill-conditioned when it exists a big disparity among its eigenvalues or characteristic speeds. Stiffness causes convergence problems regardless of the discretization method utilized, and it is measured (for one and two dimensions) by the so called condition number (Ginard et al.,

2016b),

$$\kappa = \begin{cases} \dfrac{M+1}{M}, & \text{if } M < 1/2; \\ \dfrac{M+1}{1-M}, & \text{if } 1/2 \leqslant M \leqslant 1; \\ \dfrac{M+1}{M-1}, & \text{if } M > 1. \end{cases} \tag{5.1}$$

When $M \longrightarrow 0$ or $M \longrightarrow 1$, the condition number $\kappa \longrightarrow \infty$ and the problem (2.4) becomes stiff. A strategy to reduce the disparity between the eigenvalues of the problem (2.4) and consequently decrease the condition number is the use of local preconditioning or preconditioning mass matrix schemes.

5.2 Local preconditioning for the Euler equations

Local preconditioning or preconditioning mass matrix scheme consists of premultiplying the time derivatives by a properly matrix in order to uniform the eigenvalues, smoothing the discrepancy of the different time scales. It is applied to the set of continuous equations before any discretization is done. Denoting by $\mathbf{P}$ the (nonsingular) preconditioning matrix, then the system of equations (2.4) after the preconditioning process reads

$$\mathbf{P}^{-1}\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A_x}\frac{\partial \mathbf{U}}{\partial x} + \mathbf{A_y}\frac{\partial \mathbf{U}}{\partial y} = \mathbf{0}, \quad \text{in} \quad \Omega \times (0, t_f],$$

or

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{P}\mathbf{A_x}\frac{\partial \mathbf{U}}{\partial x} + \mathbf{P}\mathbf{A_y}\frac{\partial \mathbf{U}}{\partial y} = \mathbf{0}, \quad \text{in} \quad \Omega \times (0, t_f]. \tag{5.2}$$

Even the solution evolves in time differently from that of the original problem, the time derivatives go to zero and (2.4) and (5.2) will share the same steady-state solution. In order to apply the local preconditioning technique to unsteady problems, we could use the dual-time-stepping strategy described by (Lopez et al., 2012), but it is not considered in this work. In the next subsections, we described two different local preconditioners techniques: The VLR and the WSCM.

5.2.1 Van Leer-Lee-Roe preconditioner

The *Van Leer-Lee-Roe*'s (VLR) preconditioner for the Euler equations was introduced in (Leer et al., 1991; Lee, 1996) using the symmetrizing variables with the streamline coordinates. The resulting preconditioning matrix satisfies some properties as *optimality*, *accuracy*, *continuity at the sonic point*, *preservation of the decoupled entropy equation*, *positivity*, and *symmetrizability*. The VLR preconditioner is considered optimal because it equalizes the eigenvalues of the system for all Mach numbers (Colin et al., 2011). An

explicit expression for the VLR preconditioner in conservative variables (Ginard et al., 2016a) is

$$\mathbf{P}_{VLR} = \begin{bmatrix} a_1 & a_2 u & a_2 v & a_3 \\ a_4 u & a_5 uu + \tau & a_5 uv & a_6 u \\ a_4 v & a_5 uv & a_5 vv + \tau & a_6 v \\ a_7 & a_8 u & a_8 v & a_9 \end{bmatrix}, \tag{5.3}$$

where

- $a_1 = 1 + \dfrac{\tau M^2}{\beta^2} + \dfrac{1}{2}\dfrac{R}{c_v} M^2 \Big(\dfrac{\tau M^2}{\beta^2} - 1 \Big)$;
- $a_2 = \dfrac{R}{c_v}\dfrac{1}{c^2}\Big(1 - \dfrac{\tau M^2}{\beta^2}\Big) - \dfrac{\tau}{c^2 \beta^2}$, where c is the speed of sound;
- $a_3 = \dfrac{R}{c_v}\dfrac{1}{c^2}\Big(\dfrac{\tau M^2}{\beta^2} - 1\Big)$;
- $a_4 = \dfrac{\tau(1 - M^2)}{\beta^2}\Big(1 + \dfrac{1}{2}\dfrac{R}{c_v} M^2\Big) - \dfrac{1}{2}\dfrac{R}{c_v} M^2$;
- $a_5 = \dfrac{1}{\|\mathbf{u}\|_2^2}\Big(1 + \dfrac{\tau}{\beta^2} - \tau\Big) + \dfrac{1}{c^2}\Big(\dfrac{R}{c_v}\dfrac{\tau(1 - M^2)}{\beta^2} - \dfrac{\tau}{\beta^2} + \dfrac{\beta}{c_v}\Big), \quad \|\mathbf{u}\|_2 \neq 0$;
- $a_6 = \dfrac{R}{c_v}\dfrac{1}{c^2}\Big(\dfrac{\tau(M^2 - 1)}{\beta^2} - 1\Big)$;
- $a_7 = \|\boldsymbol{u}\|_2^2 \Big[\Big(\dfrac{c_v}{R} - 1 + M^2 \Big(1 - \dfrac{1}{2}\dfrac{c_v}{R}\Big)\Big)\dfrac{\tau}{\beta^2} + \dfrac{1}{4}\dfrac{R}{c_v} M^2 \Big(\dfrac{\tau M^2}{\beta} - 1 \Big) - \dfrac{1}{2} \Big]$;
- $a_8 = 1 + \Big(1 - \dfrac{c_v}{R}\Big)\dfrac{\tau}{\beta^2} + \Big(\dfrac{R}{c_v} - \dfrac{3}{2}\Big)\dfrac{\tau M^2}{\beta} + \dfrac{1}{2}\dfrac{R}{c_v} M^2 \Big(1 - \dfrac{\tau M^2}{\beta^2}\Big)$;
- $a_9 = \Big(1 - \dfrac{R}{c_v}\Big)\dfrac{\tau M^2}{\beta^2} + \dfrac{1}{2}\dfrac{R}{c_v} M^2 \Big(\dfrac{\tau M^2}{\beta^2} - 1 \Big)$;
- $R = c_p - c_v$, in which c_p and c_v are the coefficients of specific heat at constant pressure and volume;
- $\tau = \min\big\{\beta, \frac{\beta}{M}\big\} = \begin{cases} \beta, & \text{if } M < 1 \\ \frac{\beta}{M}, & \text{if } M \geqslant 1 \end{cases}$;
- $\beta = \sqrt{|1 - M^{*2}|}$;
- $M^* = \begin{cases} M, & \text{if } M \in (0, 1 - \epsilon) \cup (1 + \epsilon, +\infty) \\ 1 - \epsilon, & \text{if } M \in (1 - \epsilon, 1) \\ 1 + \epsilon, & \text{if } M \in [1, 1 + \epsilon) \end{cases}$,

the constant ϵ is set as 0.01 according to Ginard et al. (2016b).

5.2.2 Weiss-Smith/Choi-Merkle preconditioner

Colin et al. (2011) have studied a robust low speed preconditioning formulation for viscous flows, called WSCM preconditioner. This preconditioner is based on the symmetric Weiss-Smith (WS) (Weiss and Smith, 1995) and on the viscous CM (Choi and Merkle, 1993; Ginard et al., 2016a) preconditioners. An explicit expression for the WSCM preconditioner in conservative variables (Colin et al., 2011) is

$$\mathbf{P}_{WSCM} = \mathbf{I} + \alpha \begin{bmatrix} \theta & -u & -v & 1 \\ u\theta & -uu & -uv & u \\ v\theta & -uv & -vv & v \\ H\theta & -uH & -vH & H \end{bmatrix}, \tag{5.4}$$

where

$$\alpha = \frac{\gamma - 1}{c^2}\Big[(1-\delta)\epsilon - 1\Big], \tag{5.5}$$

$$\theta = \frac{1}{2}\|\boldsymbol{u}\|^2 + \delta\frac{\epsilon c^2}{(\gamma-1)[(1-\delta)\epsilon - 1]}, \tag{5.6}$$

and H is the total enthalpy. The parameter $\delta \in [0, 1]$; for $\delta = 0$, the preconditioner is the WS preconditioner, whereas for $\delta = 1$ the CM is recovered. For Euler equations, the preconditioning parameter ϵ is given by

$$\epsilon = \min\left\{1, \max\left\{M_{lim}^2, M^2, \sigma_{pgr}\frac{|\Delta p|}{\rho c^2}\right\}\right\},$$

where $M_{lim}^2 = 10^{-5}$, $\sigma_{pgr} = 2$, and $\delta = 0.5$ according to Colin et al. (2011). In our context of finite element, we define the maximum pressure variation (Δp) on the triangle as

$$|\Delta p| = \max\{|p_1 - p_2|, |p_1 - p_3|, |p_2 - p_3|\}, \tag{5.7}$$

where p_i is the pressure on the node $i = 1, 2, 3$.

Most of the work inherent to local preconditioning is based on finite volume and finite difference methods. As far as we know, in the context of finite element the only works are described as follows: Nigro and co-authors (Nigro et al., 1997; Nigro et al., 1998) apply the CM preconditioner for solving steady compressible viscous flow; Lopez et al. (2012) extended the CM preconditioner to unsteady flow problems; Ginard and co-authors (Ginard et al., 2016a; Ginard et al., 2016b) apply the CM and VLR preconditioners for solving the Euler compressible steady flow.

5.3 The NMV methods with local preconditioning

The preconditioned nonlinear multiscale viscosity methods proposed for the Euler equation consists of finding $\mathbf{U}_{hb} = \mathbf{U}_h + \mathbf{U}_b \in \mathcal{V}_{Zhb}$ with $\mathbf{U}_h \in \mathcal{V}_{Zh}$, $\mathbf{U}_b \in \mathcal{V}_b$ such that

$$\underbrace{\int_\Omega \mathbf{W}_{hb} \cdot \Big(\frac{\partial \mathbf{U}_{hb}}{\partial t} + \mathbf{PA}_x \frac{\partial \mathbf{U}_{hb}}{\partial x} + \mathbf{PA}_y \frac{\partial \mathbf{U}_{hb}}{\partial y}\Big) d\Omega}_{\text{Galerkin term}} \; +$$

$$\underbrace{\sum_{e=1}^{nel} \int_{\Omega_e} \delta_b^{\mathbf{P}} \Big(\frac{\partial \mathbf{W}_b}{\partial x} \cdot \frac{\partial \mathbf{U}_b}{\partial x} + \frac{\partial \mathbf{W}_b}{\partial y} \cdot \frac{\partial \mathbf{U}_b}{\partial y}\Big) d\Omega}_{\text{Nonlinear subgrid stabilization term}} \; +$$

$$\underbrace{\sum_{e=1}^{nel} \int_{\Omega_e} \delta_h^{\mathbf{P}} \Big(\frac{\partial \mathbf{W}_h}{\partial x} \cdot \frac{\partial \mathbf{U}_h}{\partial x} + \frac{\partial \mathbf{W}_h}{\partial y} \cdot \frac{\partial \mathbf{U}_h}{\partial y}\Big) d\Omega}_{\text{Shock-capturing term on } \nu_{Zh}} \; = \; \mathbf{0}, \quad \forall \; \mathbf{W}_{hb} \in \mathcal{V}_{0hb}, \tag{5.8}$$

where $\mathbf{W}_{hb} = \mathbf{W}_h + \mathbf{W}_b \in \mathcal{V}_{0hb}$ with $\mathbf{W}_h \in \mathcal{V}_{0h}$, $\mathbf{W}_b \in \mathcal{V}_b$. The amount of artificial viscosity, $\delta_b^{\mathbf{P}}$ and $\delta_h^{\mathbf{P}}$, are given by Eq. (2.19), (3.12), and (3.11), according to NMV methods, where the residue of the equation on Ω_e is given by

$$R(\mathbf{U}_h) = \frac{\partial \mathbf{U}_h}{\partial t} + \mathbf{PA}_x^h \frac{\partial \mathbf{U}_h}{\partial x} + \mathbf{PA}_y^h \frac{\partial \mathbf{U}_h}{\partial y}. \tag{5.9}$$

Following the same steps of the section 3.5, we arrive at a local system of differential algebraic equations analogous to the Eq. (3.16).

5.4 Numerical Results

The flow over an airfoil is an interesting problem to examine the numerical instability coming from Mach numbers variations, that occurs in the Euler equations. This subsection shows the results of a flow passing through a NACA 0012 airfoil (Fig. 5.1) at an angle of attack of 0° and inflow Mach number from 0.01 up to 2.0.

The computational domain is discretized using Delaunay triangulation through the software Gmsh (Geuzaine and Remacle, 2009). An unstructured triangular mesh of 5,606 elements and 2,886 nodes was used for the simulation, in the computational domain given by a circle centered at the $(0, 0)$ with radius 15 (Fig. 5.2). A distance is taken ahead the leading edge of the airfoil to the inflow and outflow boundaries in order to avoid numerical instabilities of reflecting waves (Beau et al., 1993; Ginard et al., 2016b). The inflow data is set up by

$$\textit{inflow} \begin{cases} \rho &= 1.0 \\ u &= 1.0 \\ v &= 0.0 \\ T &= 1.0 \end{cases}, \tag{5.10}$$

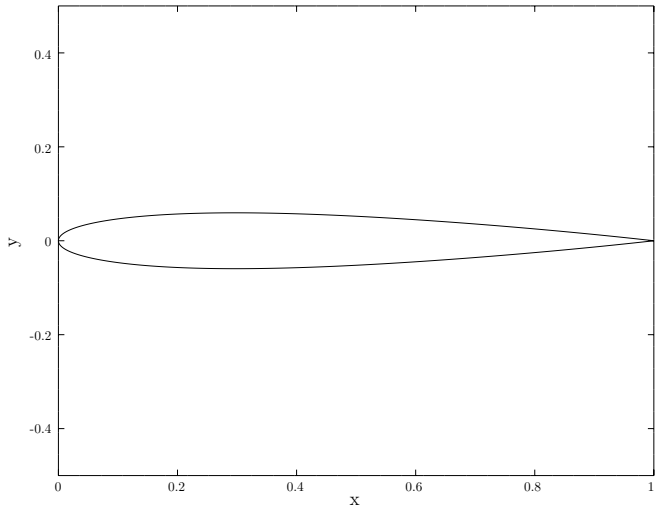
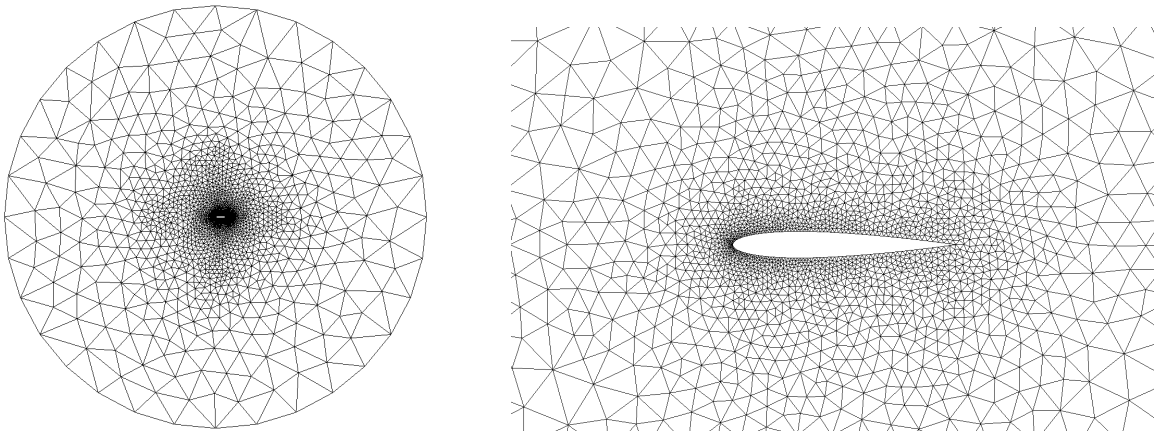


Figure 5.1 – Detail of the NACA 0012 airfoil.

where T is the temperature. Slip boundary conditions are used on the airfoil and set as described in Appendix A. As in (Ginard et al., 2016a), the coefficients c_v and c_p are set to obtain the desired inflow Mach numbers.



(a) NACA 0012 mesh (b) Detail of NACA 0012 airfoil

Figure 5.2 – Unstructured triangular mesh of 5,606 elements and 2,886 nodes.

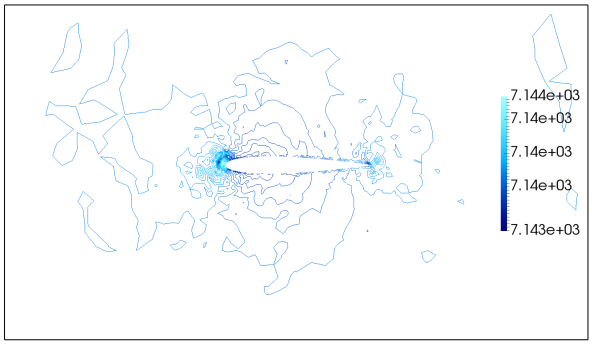
A restarted version of the GMRES solver is used to find the solution of the linearized system in each nonlinear and time iterations. The GMRES parameters are: 30 vectors to the restart process and tolerance equals 10^{-5}. The time-step size is 10^{-3} and the simulation runs until $t_f = 20.0$ (20,000 steps), and 3 fixed nonlinear iterations. For the reference values used in Eq. (2.20), we consider the inflow data given by Eq. (5.10). In this example, we evaluate the local preconditioners, WSCM and VLR, comparing them with the non-preconditioned (NP) case. The tests are performed executed employing the time integration given by Algorithm 1. We evaluate the NMV1 and NMV2 methods, which present similar solutions so that only results using the NMV1 method are shown here.

The tests are carried out with the intention of analyzing accuracy issues, specially in the incompressibility limit. Another interesting analysis would be to compare performance, once the VLR preconditioner yields optimal condition number obtainable (Colin et al., 2011). On the other hand, the WSCM preconditioner was proposed to perform robust computations. For the Euler equations, the loss of robustness occurs due to stagnation point regions, resulting in high local pressure disturbances. For the NACA 0012 airfoil simulation the $(0, 0)$ is a stagnation point, where the highest pressure values occur. Due to flow at a low speed to demonstrate an incompressible behavior, i.e., density variation is almost negligible, we use the pressure contour and energy residual to analyze this experiments.

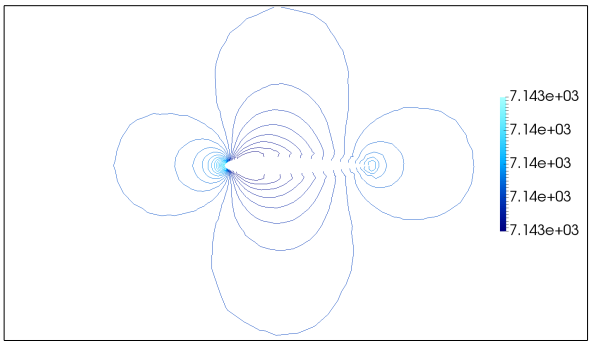
Figures 5.3-5.8 show the pressure contours for different inflow Mach numbers. The solutions are in good agreement with the solutions found in (Ginard et al., 2016a), where it is used a first order forward finite difference scheme, with the time step satisfying CFL condition. As happened in the work of (Ginard et al., 2016a) for the non-preconditioned case, our multiscale methodology does not work in the low Mach number limit, i.e., when the Mach number approaches to zero. We can see in Fig. 5.3-5.5, when $M \leqslant 0.3$, the flow at a low speed demonstrates an incompressible behavior, and methods based on conservative variables suffer with undesirable effects (Li and Xiang, 2013). The numerical solutions in the low Mach number limit are completely oscillatory, e.g. Fig. 5.3(a). On the other hand, the NMV1 method local preconditioned is able to solve problems with an incompressible behavior, as shown in Fig. 5.3(b)-(c), 5.4(b)-(c), and 5.5(b)-(c). It is worth pointing out that the non-preconditioned NMV1 becomes more stable as the Mach number increases and solutions obtained from $M = 0.3$ are comparable with the preconditioned cases.

The WSCM local preconditioner, as far as we know, was applied in finite volume context for low speed regime. Even when defined for low speed regime, the WSCM presents comparable results to the optimal local preconditioner, VLR, for all regimes simulated in this work, inflow Mach numbers: 0.01; 0.1; 0.3; 0.5; 1.0; 2.0.

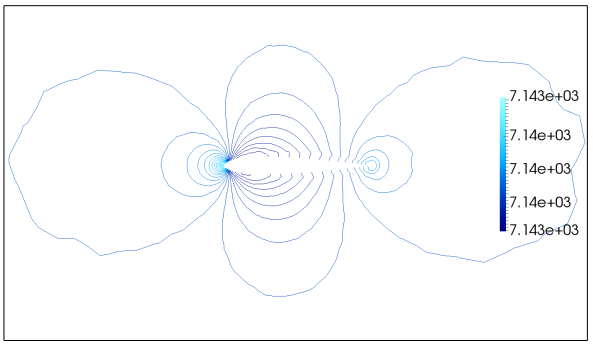
Figure 5.9 shows the time evolution of the energy residual $L^2(\Omega)$-norm. The residual norm of the NMV1(NP) for $M = 0.01$ oscillates between 10^0 and 10^{-1}, remains approximately constant for $M = 0.1$ and for others Mach numbers (Fig. 5.9(c)-(f)) decreasing from 10^{-1} up to 10^{-2}. For the NMV1 method combined with the VLR and WSCM preconditioners the energy residual sequence decay from 10^{-3} up to 10^{-4} in fewer than 20,000 steps in the incompressible limit, as we can see in Fig. 5.9(a)-(c). In transonic case, $M = 1.0$ (Fig. 5.9(e)), the residual of the NMV1(WSCM) method shows greater decay in relation to the NMV1(VLR) method, we can also observe in Fig. 5.7 that the solution NMV1(VLR) exhibit more oscillations than the NMV1(WSCM) one. In supersonic case, $M = 2.0$, the residual sequence of the NMV1(VLR) shows a decreasing from 10,000 steps, whereas the residual sequence of the NMV1(WSCM) remaining approximately constant



(a) NMV1(NP)



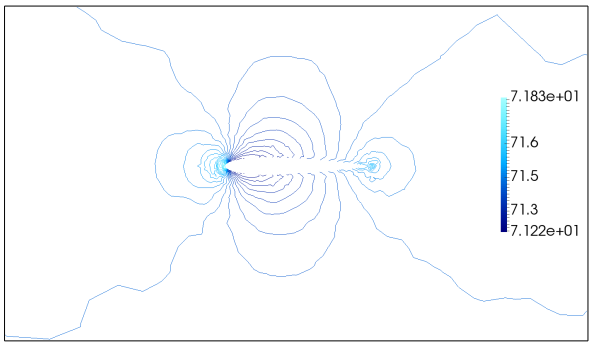
(b) NMV1(VLR)



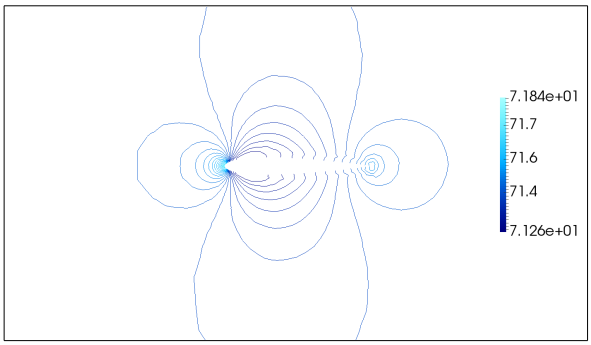
(c) NMV1(WSCM)

Figure 5.3 – NACA 0012: Pressure contours for $M = 0.01$ at the inflow.

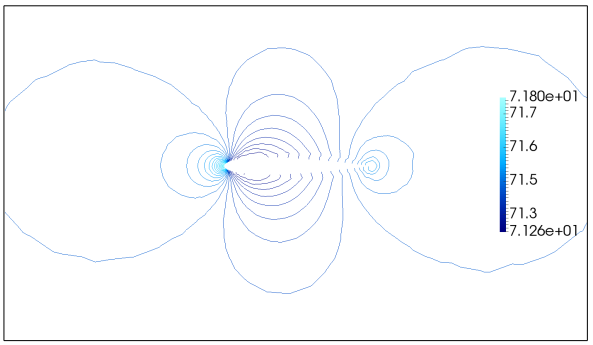
around 10^{-3}, Fig. 5.9(f).



(a) NMV1(NP)

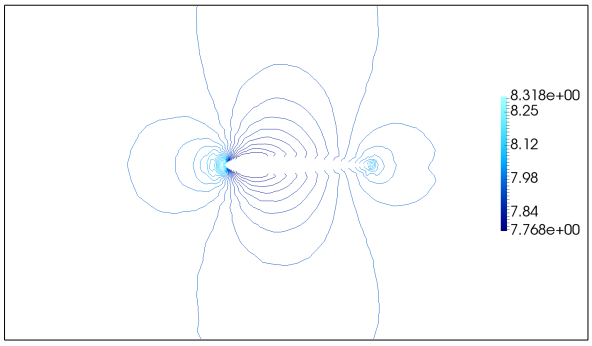


(b) NMV1(VLR)

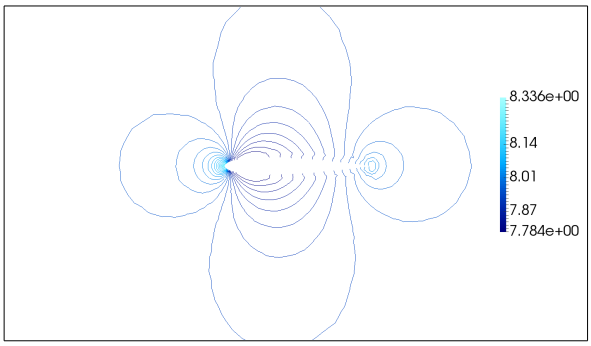


(c) NMV1(WSCM)

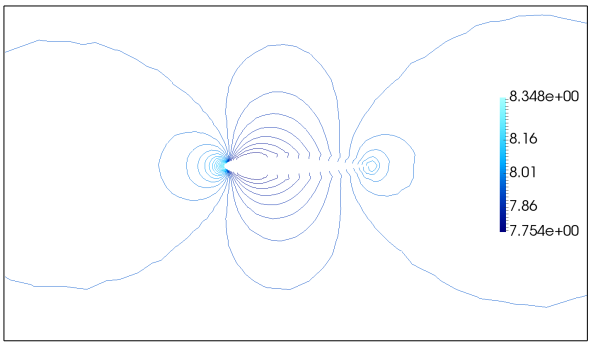
Figure 5.4 – NACA 0012: Pressure contours for $M = 0.1$ at the inflow.



(a) NMV1(NP)

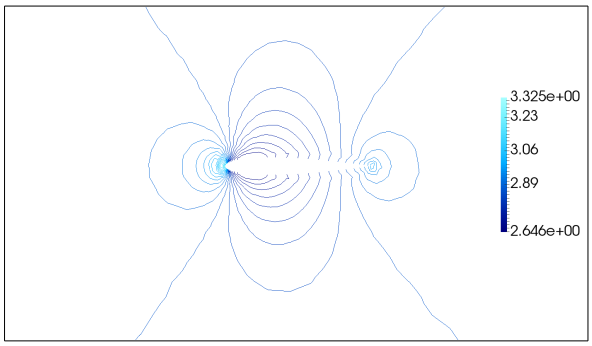


(b) NMV1(VLR)

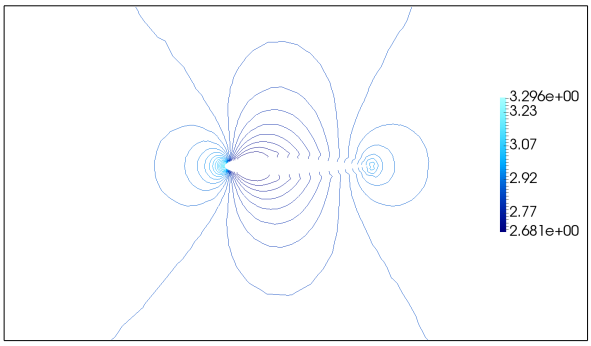


(c) NMV1(WSCM)

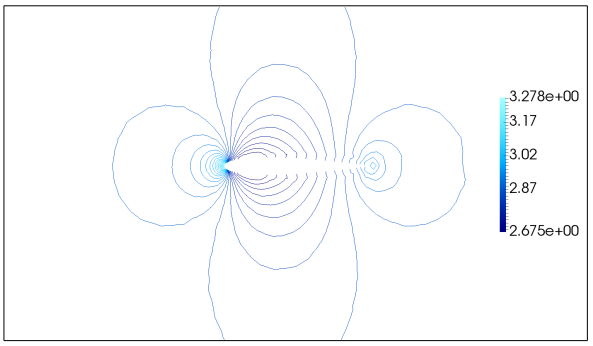
Figure 5.5 – NACA 0012: Pressure contours for $M = 0.3$ at the inflow.



(a) NMV1(NP)

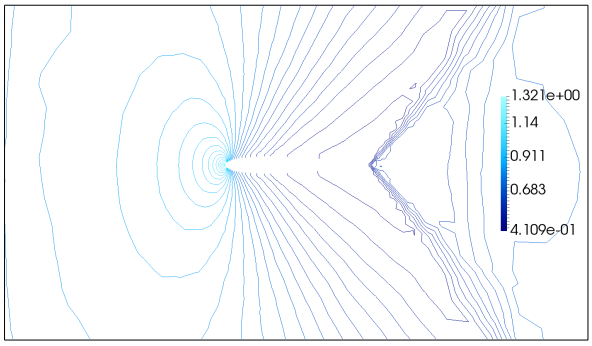


(b) NMV1(VLR)

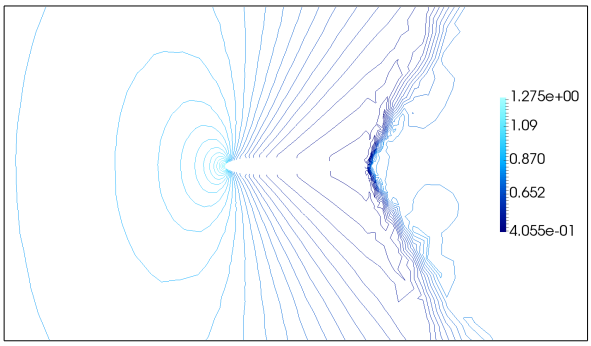


(c) NMV1(WSCM)

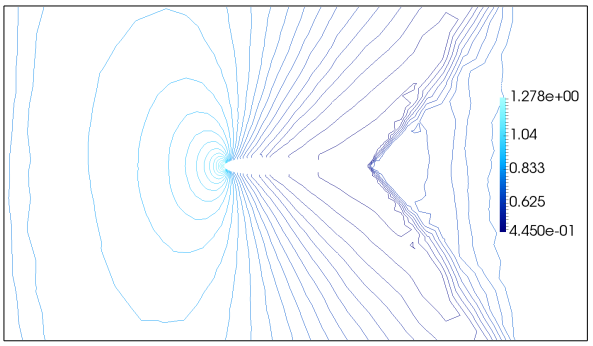
Figure 5.6 – NACA 0012: Pressure contours for $M = 0.5$ at the inflow.



(a) NMV1(NP)

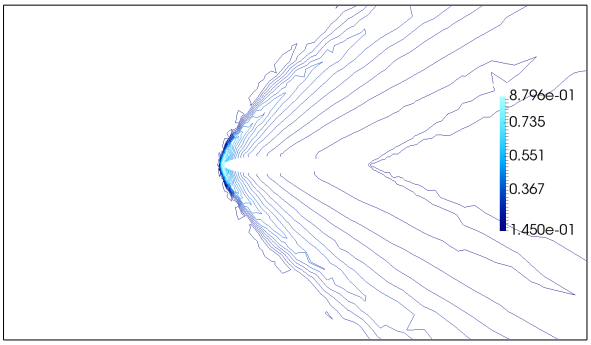


(b) NMV1(VLR)

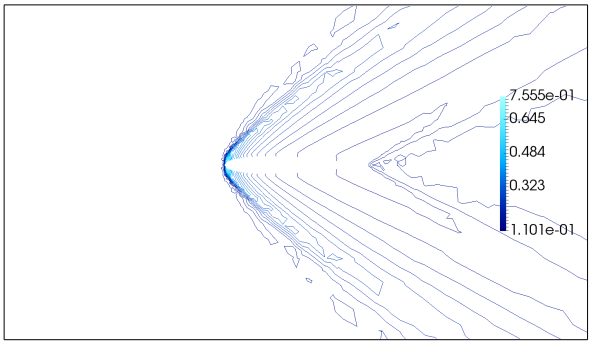


(c) NMV1(WSCM)

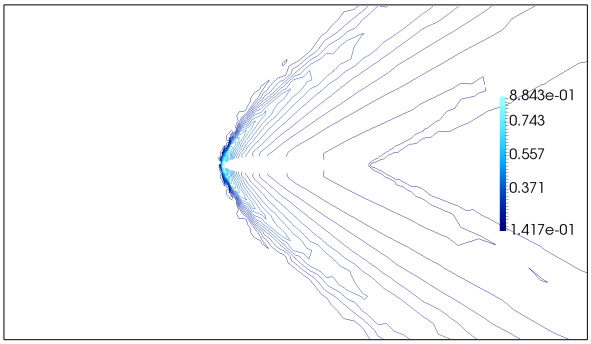
Figure 5.7 – NACA 0012: Pressure contours for $M = 1.0$ at the inflow.



(a) NMV1(NP)

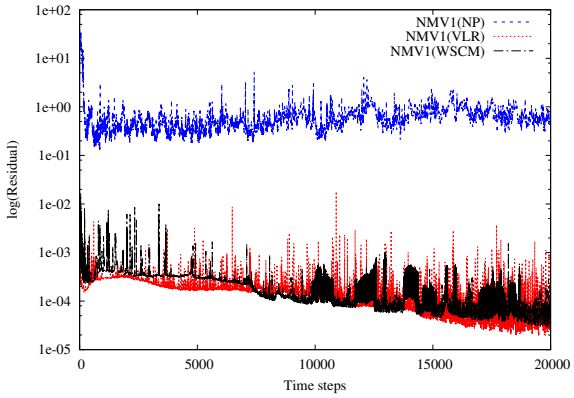


(b) NMV1(VLR)

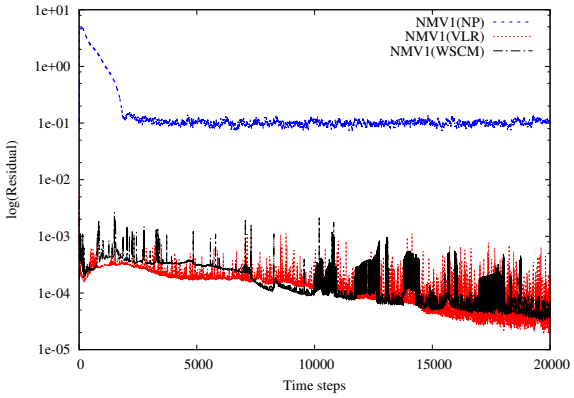


(c) NMV1(WSCM)

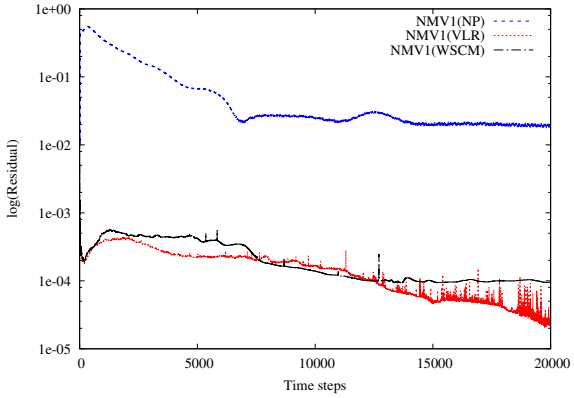
Figure 5.8 – NACA 0012: Pressure contours for $M = 2.0$ at the inflow.



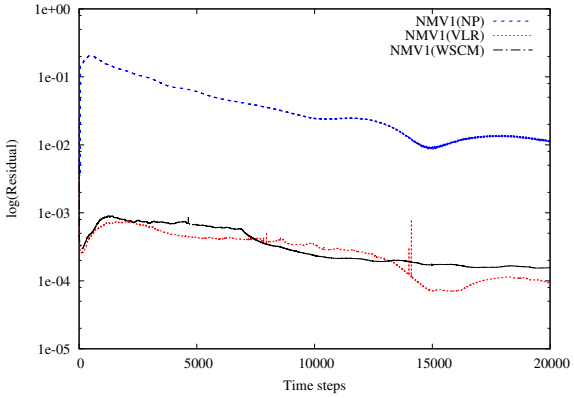
(a) Mach = 0.01



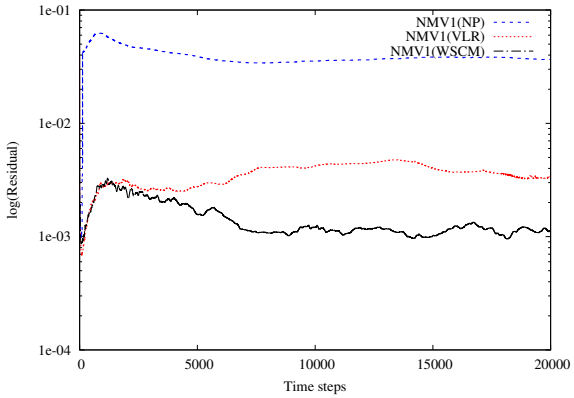
(b) Mach = 0.1



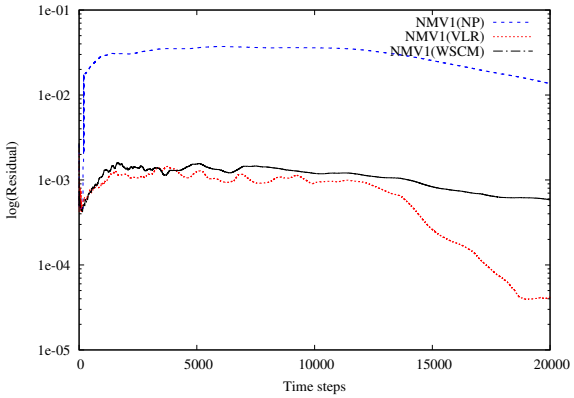
(c) Mach = 0.3



(d) Mach = 0.5



(e) Mach = 1.0



(f) Mach = 2.0

Figure 5.9 – NACA 0012 problem: Residual of energy.

6 Conclusions

6.1 Review of results

In this work we present two new nonlinear multiscale finite element formulations to solve inviscid compressible flows in conservative variables. These formulations called NMV1 and the NMV2 methods are self adaptive and parameter-free. In the NMV1 formulation an artificial nonlinear viscosity is added isotropically on all scales of the discretization, following the same philosophy of the DD method. On the other hand, in the NMV2 formulation, different nonlinear operators are included on macro and micro scales. In both formulations, the amount of artificial viscosity is inspired in the YZβ shock-capturing viscosity parameter.

Solutions obtained with the NMV1 and the NMV2 methods are comparable with those obtained with the SUPG + YZβ method in all experiments – from subsonic to supersonic flow, and in some cases the NMV methods yields solutions slightly more accurate, as in the shock tube, explosion, and tunnel examples. Furthermore, the NMV methods improve the conditioning of the resulting linear system requiring fewer GMRES iterations, as can be seen in all computational experiments shown in Section 3.6. This makes substantial difference in the computational time.

Another topic studied in this thesis is time integration schemes. We introduce a different time integration scheme to solve the resulting system of differential algebraic equations (DAE) originated from the spatial discretization of the NMV models. It is well known that temporal instabilities can appear in the numerical solutions of stiff equations, as Euler equations. A strategy extremely fast at dissipating numerical instabilities in stiff equations is to use BDF methods having the decay stiff property, for example, the BDF-2 method. It is worth stressing that classical prediction-correction methods are based on explicit methods in the prediction phase and implicit methods in the correction phase. According to Quarteroni et al. (2006), these methods present the stability property of the methods used in the prediction step, i.e., the classical predictor-corrector methods present the stability of explicit methods. The numerical scheme proposed here, called PC-BDF2, is a BDF-2-based predictor-corrector where the prediction phase is done in a non standard way, through of a extrapolation mechanism, maintaining the decay stiff stability property of the implicit BDF-2 method. This method is designed to solve problems in the multiscale variational context, allowing both scales evolve in time. The PC-BDF2 method was compared with the standard predictor-correct method, called PC-std, in the solution of two transonic problems (shock tube and explosion) through the NMV methods. The PC-BDF2 generally needed just one nonlinear iteration to satisfy the error tolerance

whereas the PC-std does not satisfy the nonlinear error tolerance, spending the maximum number of iterations. Thus, the PC-BDF2 scheme needs fewer GMRES iterations and fewer nonlinear corrections than the PC-std scheme. Moreover, the PC-BDF2 solutions are slightly more accurate.

Finally, another contribution of this thesis relies in the local preconditioning in order to deal with the stiffness property of compressible flow, specially when the Mach number approaches to zero. In this case, methods based on conservative variables fail. Considering only steady state problems, the NMV1 method combined with two local preconditioners, VLR and WSCM, was applied in the NACA 0012 airfoil problem for the incompressible flow limit. We simulate the flow over the NACA 0012 airfoil under various regimes of inflow Mach numbers: 0.01; 0.1; 0.3; 0.5; 1.0; 2.0. The solutions obtained with the NMV1 without local preconditioning are completely oscillatory in the low Mach number limit (0.01; 0.1 and 0.3). On the other hand, the NMV1 method combined with local preconditioning presents good results, as shown in Section 5.4.

6.2 Future works

In this thesis we propose a class of nonlinear multiscale variational methods (NMV1 and NMV2) to solve compressible flow and a predictor-corrector time integration scheme based on BDF-2 in the multiscale variational context (PC-BDF2), allowing both scales evolve in time. Moreover, considering only steady state problems, the NMV1 method combined with two local preconditioners, VLR and WSCM, was applied in the NACA 0012 airfoil problem for the incompressible flow limit. There are several lines of research arising from this work which should be pursued, such that,

- Development of a numerical analysis for the NMV methods. We can explorer stability issues using Fourier analysis;
- Extension of these methodologies for solving 3D Euler equations and the compressible Navier-Stokes equations;
- Extension of these methodologies for the discontinuous Galerkin framework;
- Development of a temporal numerical analysis for the PC-BDF2 method. In our numerical experiment the PC-BDF2 achieved an order of convergence greater than 2, characterizing a hyper-convergence effect. Thus, a temporal numerical analysis must be done to clearly understand this phenomenon;
- Application of the local preconditioning technique in the solution of unsteady problems. In this work we apply the local preconditioning for solving only steady

flow problems with the NMV1 method. The solution of unsteady problems is done through the dual-time-stepping technique, as proposed in (Lopez et al., 2012).

Bibliography

ABBASSI, H.; MASHAYEK, F.; JACOBS, G. B. Shock capturing with entropy-based artificial viscosity for staggered grid discontinuous spectral element method. *Computers & Fluids*, v. 98, p. 152–163, 2014.

ALMEIDA, R. C.; GALEÃO, A. C. An adaptive Petrov-Galerkin formulation for the compressible Euler and Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, v. 129, n. 1, p. 157 – 176, 1996.

ANDERSON, J. *Modern Compressible Flow: With Historical Perspective*. McGraw-Hill Education, 2003. (Aeronautical and Aerospace Engineering Series).

ARRUDA, N. C. B.; ALMEIDA, R. C.; CARMO, E. G. D. do. Dynamic diffusion formulations for advection dominated transport problems. *Mecánica Computacional*, v. 29, p. 2011–2025, 2010.

ASHINO, R.; NAGASE, M.; VAILLANCOURT, R. Behind and beyond the MATLAB ODE suite. *Computers and Mathematics with Applications*, v. 40, n. 4, p. 491–512, 2000.

BASSI, F.; BARTOLO, C. D.; HARTMANN, R.; NIGRO, A. A discontinuous Galerkin method for inviscid low Mach number flows. *Journal of Computational Physics*, v. 228, n. 11, p. 3996 – 4011, 2009.

BEAU, G. L.; RAY, S.; ALIABADI, S.; TEZDUYAR, T. SUPG finite element computation of compressible flows with the entropy and conservation variables formulations. *Computer Methods in Applied Mechanics and Engineering*, v. 104, n. 3, p. 397 – 422, 1993.

BENTO, S. S.; BARBOSA, P. W.; SANTOS, I. P.; LIMA, L. M. de; CATABRIGA, L. A nonlinear finite element formulation based on multiscale approach to solve compressible Euler equations. In: *Computational Science and Its Applications - ICCSA 2017 - 17th International Conference, Trieste, Italy, July 3-6, 2017, Proceedings, Part VI*. 2017. p. 735–743.

BENTO, S. S.; LIMA, L. M. de; SEDANO, R. Z.; CATABRIGA, L.; SANTOS, I. P. A nonlinear multiscale viscosity method to solve compressible flow problems. In: *Computational Science and Its Applications - ICCSA 2016 - 16th International Conference, Beijing, China, July 4-7, 2016, Proceedings, Part I*. 2016. p. 3–17.

BRENNER, S. C.; SCOTT, L. R. *The mathematical theory of finite element methods*. New York, Berlin, Paris: Springer, 2002. (Texts in applied mathematics).

BRILEY, W.; TAYLOR, L.; WHITFIELD, D. High-resolution viscous flow simulations at arbitrary Mach number. *Journal of Computational Physics*, v. 184, n. 1, p. 79 – 105, 2003.

BROOKS, A. N.; HUGHES, T. J. R. Streamline upwind Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, v. 32, p. 199–259, 1982.

CASH, J. R. Review paper: Efficient numerical methods for the solution of stiff initial-value problems and differential algebraic equations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, v. 459, n. 2032, p. 797–815, 2003.

CATABRIGA, L.; COUTINHO, A. L. Implicit SUPG solution of Euler equations using edge-based data structures. *Computer Methods in Applied Mechanics and Engineering*, v. 191, n. 32, p. 3477 – 3490, 2002.

CATABRIGA, L.; SOUZA, D. A. F. de; COUTINHO, A. L. G. A.; TEZDUYAR, T. E. Three-dimensional edge-based SUPG computation of inviscid compressible flows with YZβ shock-capturing. *Journal of Applied Mechanics*, v. 76, n. 2, p. 021208–021208–7, 2009.

CHOI, Y.-H.; MERKLE, C. The application of preconditioning in viscous flows. *Journal of Computational Physics*, v. 105, n. 2, p. 207 – 223, 1993.

CHORIN, A. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, v. 2, n. 1, p. 12 – 26, 1965.

COLIN, Y.; DENIAU, H.; BOUSSUGE, J.-F. A robust low speed preconditioning formulation for viscous flow computations. *Computers & Fluids*, v. 47, n. 1, p. 1–15, 2011.

CURTISS, C. F.; HIRSCHFELDER, J. O. Integration of stiff equations. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, v. 38, n. 3, p. 235–243, 1952.

DIONE, I.; URQUIZA, J. M. Penalty: finite element approximation of stokes equations with slip boundary conditions. *Numerische Mathematik*, v. 129, n. 3, p. 587–610, 2015.

EMERY, A. F. An evaluation of several differencing methods for inviscid fluid flow problems. *Journal of Computational Physics*, v. 2, n. 3, p. 306 – 331, 1968.

FÖRSTER, C. *Robust methods for fluid-structure interaction with stabilised finite elements*. Thesis (PhD) — Institut für Baustatik und Baudynamik, Universität Stuttgart, Germany, 2007.

FOX, R. W.; MCDONALD, A. T.; PRITCHARD, P. J. Introduction to fluid dynamics. *John Wiley & Sons*, 2004.

FRANCA, L.; NESLITURK, A.; STYNES, M. On the stability of residual-free bubbles for convection-diffusion problems and their approximation by a two-level finite element method. *Computer Methods in Applied Mechanics and Engineering*, v. 166, n. 1, p. 35 – 49, 1998.

FRANCA, L. P.; FARHAT, C. Bubble functions prompt unusual stabilized finite element methods. *Computer Methods in Applied Mechanics and Engineering*, v. 123, n. 1, p. 299 – 308, 1995.

GALEÃO, A. C.; CARMO, E. G. D. do. A consistent approximate upwind Petrov-Galerkin method for convection-dominated problems. *Computer Methods in Applied Mechanics and Engineering*, v. 68, p. 83–95, 1988.

GEAR, C. W. *Numerical initial value problems in ordinary differential equations*. Prentice Hall PTR, 1971.

GEUZAINE, C.; REMACLE, J.-F. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, v. 79, n. 11, p. 1309–1331, 2009.

GINARD, M. M.; BERNARDINO, G.; VÁZQUEZ, M.; HOUZEAUX, G. Fourier stability analysis and local Courant number of the preconditioned variational multiscale stabilization (P-VMS) for Euler compressible flow. *Computer Methods in Applied Mechanics and Engineering*, v. 301, p. 28 – 51, 2016.

GINARD, M. M.; VÁZQUEZ, M.; HOUZEAUX, G. Local preconditioning and variational multiscale stabilization for Euler compressible steady flow. *Computer Methods in Applied Mechanics and Engineering*, v. 305, p. 468 – 500, 2016.

GRAEBEL, W. *Advanced fluid mechanics*. Academic Press, 2007.

GRAVEMEIER, V.; GEE, M. W.; KRONBICHLER, M.; WALL, W. A. An algebraic variational multiscale–multigrid method for large eddy simulation of turbulent flow. *Computer Methods in Applied Mechanics and Engineering*, v. 199, n. 13, p. 853–864, 2010.

GUERMOND, J. L. Stabilization of Galerkin approximations of transport equations by subgrid modeling. *ESAIM: Mathematical Modelling and Numerical Analysis*, v. 33, n. 06, p. 1293–1316, 1999.

GUERMOND, J. L. Subgrid stabilization of Galerkin approximations of linear monotone operators. *IMA Journal of Numerical Analysis*, v. 21, n. 1, p. 165–197, 2001.

HAIRER, E.; NRSETT, S. P.; WANNER, G. *Solving Ordinary Differential Equations: Nonstiff problems. v. 2: Stiff and differential-algebraic problems*. Springer Verlag, 2010.

HAIRER, E.; WANNER, G. *Solving ordinary differential equations II: Stiff and differential-algebraic problems second revised edition with 137 figures*. Springer-Verlag, 1996. v. 14.

HAIRER, E.; WANNER, G. Linear multistep method. *Scholarpedia*, v. 5, n. 4, p. 4591, 2010. Revision #91436.

HAY, A.; ETIENNE, S.; PELLETIER, D.; GARON, A. hp-Adaptive time integration based on the BDF for viscous flows. *Journal of Computational Physics*, Elsevier, v. 291, p. 151–176, 2015.

HUEBNER, K. H.; DEWHIRST, D. L.; SMITH, D. E.; BYROM, T. G. *The finite element method for engineers*. Fourth. John Wiley & Sons, 2001.

HUGHES, T. J. R. Multiscale phenomena: Green's functions, the Dirichlet-to-Neumann formulation, sugrid scale models, bubbles and the origin of stabilized methods. *Computer Methods in Applied Mechanics and Engineering*, v. 127, p. 387–401, 1995.

HUGHES, T. J. R.; SCOVAZZI, G.; FRANCA, L. P. Multiscale and stabilized methods. *Encyclopedia of Computational Mechanics*, 2004.

HUGHES, T. J. R.; SCOVAZZI, G.; TEZDUYAR, T. E. Stabilized methods for compressible flows. *Journal of Scientific Computing*, v. 43, n. 3, p. 343–368, 2010.

HUGHES, T. J. R.; TEZDUYAR, T. E. Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible Euler equations. *Computer methods in applied mechanics and engineering*, v. 45, n. 1, p. 217–284, 1984.

JOHN, V.; KNOBLOCH, P. On spurious oscillations at layers diminishing (SOLD) methods for convection–diffusion equations: Part i–a review. *Computer Methods in Applied Mechanics and Engineering*, v. 196, n. 17, p. 2197–2215, 2007.

KNOLL, D.; KEYES, D. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, v. 193, n. 2, p. 357 – 397, 2004.

LEE, D. *Local Preconditioning of the Euler and Navier-Stokes Equations*. Thesis (PhD) — University of Michigan, 1996.

LEE, D. Design criteria for local Euler preconditioning. *Journal of Computational Physics*, Elsevier, v. 144, n. 2, p. 423–459, 1998.

LEER, B. V.; LEE, W.-T.; ROE, P. L. Characteristic time-stepping or local preconditioning of the Euler equations. American Institute of Aeronautics and Astronautics, 1991.

LI, Z.; XIANG, H. Development of a Navier-Stokes flow solver for all speeds on unstructured grids. *Engineering Letters*, v. 21, n. 2, p. 89–94, 2013.

LOPEZ, E. J.; NIGRO, N. M.; SARRAF, S. S.; DAMIÁN, S. M. Stabilized finite element method based on local preconditioning for unsteady compressible flows in deformable domains with emphasis on the low Mach number limit application. *International Journal for Numerical Methods in Fluids*, John Wiley & Sons, Ltd, v. 69, n. 1, p. 124–145, 2012.

MATTOS, R. N. *Formulações Estabilizadas Multiescalas Aplicadas às Equações de Euler*. Dissertation (Msc) — UFES, Vitória, ES, Agosto 2012.

MITTAL, S.; TEZDUYAR, T. A unified finite element formulation for compressible and incompressible flows using augmented conservation variables. *Computer Methods in Applied Mechanics and Engineering*, v. 161, n. 3, p. 229 – 243, 1998.

NASSEHI, V.; PARVAZINIA, M. A multiscale finite element space–time discretization method for transient transport phenomena using bubble functions. *Finite Elements in Analysis and Design*, v. 45, n. 5, p. 315–323, 2009.

NAZAROV, M.; LARCHER, A. Numerical investigation of a viscous regularization of the Euler equations by entropy viscosity. *Computer Methods in Applied Mechanics and Engineering*, v. 317, p. 128 – 152, 2017.

NIGRO, N.; STORTI, M.; IDELSOHN, S. GMRES physics-based preconditioner for all Reynolds and Mach numbers: numerical examples. *International Journal for Numerical Methods in Fluids*, John Wiley & Sons, Ltd, v. 25, n. 12, p. 1347–1371, 1997.

NIGRO, N.; STORTI, M.; IDELSOHN, S.; TEZDUYAR, T. Physics based GMRES preconditioner for compressible and incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, v. 154, n. 3, p. 203 – 228, 1998.

NOELLE, S.; BISPEN, G.; ARUN, K.; LUKÁCOVÁ-MEDVID'OVÁ, M.; MUNZ, C. A weakly asymptotic preserving low mach number scheme for the euler equations of gas dynamics. *SIAM Journal on Scientific Computing*, v. 36, n. 6, p. B989–B1024, 2014.

QUARTERONI, A.; SACCO, R.; SALERI, F. *Numerical Mathematics (Texts in Applied Mathematics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

QUARTERONI, A.; SALERI, F. *Scientific Computing with MATLAB and Octave (Texts in Computational Science and Engineering)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

RISPOLI, F.; SAAVEDRA, R.; CORSINI, A.; TEZDUYAR, T. E. Computation of inviscid compressible flows with the V-SGS stabilization and YZβ shock-capturing. *International Journal for Numerical Methods in Fluids*, v. 54, n. 6-8, p. 695–706, 2007.

SANTOS, I. P.; ALMEIDA, R. C. A nonlinear subgrid method for advection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, v. 196, p. 4771–4778, 2007.

SANTOS, I. P.; ALMEIDA, R. C.; MALTA, S. M. C. Numerical analysis of the nonlinear subgrid scale method. *Computational & Applied Mathematics*, v. 31, n. 3, p. 473–503, 2012.

SEDANO, R. Z.; BENTO, S. S.; LIMA, L. M.; CATABRIGA, L. Predictor-multicorrector schemes for the multiscale dynamic diffusion method to solve compressible flow problems. *CILAMCE2015 - XXXVI Ibero-Latin American Congress on Computational Methods in Engineering.*, nov. 2015.

SHAKIB, F.; HUGHES, T. J. R.; JOHAN, Z. A new finite element formulation for computational fluid dynamics: X. the compressible Euler and Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, Elsevier, v. 89, n. 1, p. 141–219, 1991.

SOD, G. A. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of computational physics*, Elsevier, v. 27, n. 1, p. 1–31, 1978.

SÖDERLIND, G.; JAY, L.; MANUEL, C. Stiffness 1952-2012. sixty years in search of a definition. *BIT*, Springer, v. 55, n. 2, p. 531–558, 2015.

TEZDUYAR, T. E. Adaptive determination of the finite element stabilization parameters. *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS*, 2001.

TEZDUYAR, T. E. Determination of the stabilization and shock-capturing parameters in supg formulation of compressible flows. *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS*, 2004.

TEZDUYAR, T. E. Finite elements in fluids: Stabilized formulations and moving boundaries and interfaces. *Computers and Fluids*, v. 36, n. 2, p. 191–206, 2007.

TEZDUYAR, T. E.; SENGA, M. Stabilization and shock-capturing parameters in SUPG formulation of compressible flows. *Computer Methods in Applied Mechanics and Engineering*, v. 195, n. 13-16, p. 1621–1632, 2006.

TEZDUYAR, T. E.; SENGA, M. SUPG finite element computation of inviscid supersonic flows with YZβ shock-Capturing. *Computers & Fluids*, n. 1, p. 147–159, 2007.

TORO, E. F. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2009.

TURKEL, E. Preconditioned methods for solving the incompressible and low speed compressible equations. *Journal of Computational Physics*, v. 72, p. 277 – 298, 1987.

VALLI, A. M.; ALMEIDA, R. C.; SANTOS, I. P.; CATABRIGA, L.; MALTA, S. M.; COUTINHO, A. L. A parameter-free dynamic diffusion method for advection–diffusion–reaction problems. *Computers & Mathematics with Applications*, p. 307–321, 2017.

VALLI, A. M. P.; CATABRIGA, L.; SANTOS, I. P.; COUTINHO, A. L. G. A.; ALMEIDA, R. C. Time integration schemes for a multiscale finite element formulation to solve transient advection-diffusion-reaction problems. *CILAMCE2014 - XXXVI Ibero-Latin American Congress on Computational Methods in Engineering.*, 2014.

VALLI, A. M. P.; CATABRIGA, L.; SANTOS, I. P.; COUTINHO, A. L. G. A.; ALMEIDA, R. C. Predictor-multicorrector scheme for the dynamic diffusion method. *CMAC-SE - Congresso de Matemática Aplicada e Computacional do Sudeste*, 2015.

VENKATESWARAN, S.; MERKLE, C. L. Analysis of preconditioning methods for the Euler and Navier-Stokes equations. *Lecture Series - Van Karman Institute for Fluid Dynamics*, v. 3, p. B1–B155, 1999.

WEISS, J.; SMITH, W. Preconditioning applied to variable and constant density flows. *AIAA Journal*, v. 33, n. 11, p. 2050 – 2057, 1995.

WERNER, S. L.; CATABRIGA, L.; SANTOS, I. P. Métodos de estabilização submalha difusão dinâmica aplicado na simulação de escoamento miscível. *Mecánica Computacional*, v. 29, p. 4039–4053, 2010.

WONG, J.; DARMOFAL, D.; PERAIRE, J. The solution of the compressible Euler equations at low Mach numbers using a stabilized finite element algorithm. *Computer Methods in Applied Mechanics and Engineering*, v. 190, n. 43, p. 5719 – 5737, 2001.

WOODWARD, P.; COLELLA, P. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics*, Elsevier, v. 54, n. 1, p. 115–173, 1984.

APPENDIX A – Boundary condition

Generally, natural and essential boundary condition are simple to implement, whereas the non-penetration boundary condition, where the resulting normal velocity field in relation to the impermeable wall should disappear, it is not easy to apply. Mathematically, the no-penetration boundary condition is given by

$$\boldsymbol{u} \cdot \boldsymbol{n} = 0, \quad \text{on} \quad \Gamma_{\text{wall}} \times (0, T_f), \tag{A.1}$$

where $\boldsymbol{n}$ is the external unit normal to the wall. This condition is simple to apply on vertical or horizontal walls, but on arbitrary geometry boundaries, the imposition of this condition turn into a nontrivial task. In this case it is necessary to apply some strategy to impose the boundary condition.

A.1 Weak non-penetration boundary condition

The weak imposition of the slip boundary condition is enforced adding a penalty term to the variational equation (Dione and Urquiza, 2015; Nazarov and Larcher, 2017), as follow

$$\frac{1}{\varepsilon} \int_{\Gamma_{\text{wall}}} (\boldsymbol{u}_h \cdot \boldsymbol{n}_h)(\boldsymbol{v}_h \cdot \boldsymbol{n}_h) d\Gamma, \tag{A.2}$$

where $\varepsilon > 0$ is the penalty parameter, also referred to as a penetration constant. When $\varepsilon \to 0$ the solid boundary becomes completely impermeable. However, smaller values of ε may cause restriction to the time step size for explicit methods (Nazarov and Larcher, 2017).

A.2 Strong non-penetration boundary condition

The strong imposition of the slip boundary condition requires algebraic manipulations in the linear system, mapping the nodes on the solid wall and recombining linearly the components of the velocity so as to satisfy the desired boundary condition. A strategy to enforce the slip boundary condition strongly consists of a coordinate transformation. It may be more convenient to rewrite the matrix or vector of unknowns on the node in a local coordinate system – a coordinate system associated with the node.

We can write the transformation equation (Huebner et al., 2001), for nodal matrix and nodal vector of unknowns, as

$$\boldsymbol{K}^i = \boldsymbol{R}_i^T \boldsymbol{K}_L^i \boldsymbol{R}_i \quad \text{and} \quad \boldsymbol{U}_h^i = \boldsymbol{R}_i \boldsymbol{U}_L^i, \tag{A.3}$$

where the subscript L denotes local coordinates, $\boldsymbol{R}_i$ is the transformation (rotation) matrix and i is the element node. The rotation matrix is defined as

$$\mathbf{R}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_i & -\sin\theta_i & 0 \\ 0 & \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{A.4}$$

where θ_i $(i = 1, 2, 3)$ is the angle measured from the x-axis and the local tangent axis to the solid wall on the node i. Using the coordinate rotation, we can set the normal velocity component to zero in the local coordinate system as Dirichlet boundary condition.

In this work, as in Beau et al. (1993), the velocity unknowns on the solid wall are setting in local coordinate system, allowing the degree of freedom of velocity to be specified as a Dirichlet boundary condition. Then the nodal vector of unknowns on the solid wall are rotated to a global Cartesian coordinate system. The operational details to apply this boundary condition in our methodology are presented below.

A.3 Coordinate rotation applied to the NMV methods

The approximated solutions $\mathbf{U}^{e,h}$ and $\mathbf{U}^{e,b}$, by finite element discretization, within each element is given by

$$\mathbf{U}_h^e = N_1 \begin{bmatrix} U_1^1 \\ U_2^1 \\ U_3^1 \\ U_4^1 \end{bmatrix} + N_2 \begin{bmatrix} U_1^2 \\ U_2^2 \\ U_3^2 \\ U_4^2 \end{bmatrix} + N_3 \begin{bmatrix} U_1^3 \\ U_2^3 \\ U_3^3 \\ U_4^3 \end{bmatrix}, \tag{A.5}$$

$$\mathbf{U}_b^e = 27 N_1 N_2 N_3 \begin{bmatrix} U_1^b \\ U_2^b \\ U_3^b \\ U_4^b \end{bmatrix}, \tag{A.6}$$

where N_j is the local shape function associated with node $j = 1, 2, 3$.

The element matrix of the NMV methods, before condensation, is a matrix of order 16, as follow

$$\mathbf{A}^e = \begin{array}{c} \begin{array}{cccc} \text{node 1} & \text{node 2} & \text{node 3} & \text{node 4} \end{array} \\ \left[\begin{array}{ccc|c} \mathbf{A}_{11}^e & \mathbf{A}_{12}^e & \mathbf{A}_{13}^e & \mathbf{A}_{14}^e \\ \mathbf{A}_{21}^e & \mathbf{A}_{22}^e & \mathbf{A}_{23}^e & \mathbf{A}_{24}^e \\ \mathbf{A}_{31}^e & \mathbf{A}_{32}^e & \mathbf{A}_{33}^e & \mathbf{A}_{34}^e \\ \hline \mathbf{A}_{41}^e & \mathbf{A}_{42}^e & \mathbf{A}_{43}^e & \mathbf{A}_{44}^e \end{array}\right] \begin{array}{l} \text{node 1} \\ \text{node 2} \\ \text{node 3} \\ \text{node 4} \end{array} \end{array} \tag{A.7}$$

where each one of the sub-matrices, $\mathbf{A}_{ij}^{e}$ com $1 \leqslant i, j \leqslant 4$, has order 4. On the nodes 1, 2, and 3 the resolved solution is calculated. On the other hand, on the node 4, at the barycenter of the triangular element, the unresolved solution is obtained.

Equation (A.5) uses a nodal vector of unknowns based on a global Cartesian coordinate system. In order to enforce the slip boundary condition using coordinate rotation, a new nodal vector of unknowns for solid wall nodes is defined where the second and third components of this new vector represent components of the velocity in the tangential ($\boldsymbol{t}$) and normal ($\boldsymbol{n}$) directions, i.e.,

$$\mathbf{U}_{L}^{i} = \begin{bmatrix} \rho \\ \rho u_t \\ \rho u_n \\ \rho E \end{bmatrix}, \tag{A.8}$$

where the superscript i refers to the local node.

Considering that only nodes 1, 2 and 3 may be in the boundary, suppose, without loss of generality, that node 1 lying on the impermeable boundary, Eq. (A.5) would be modified for the node 1, such that

$$\mathbf{U}_{h}^{e} = N_1 \mathbf{R}_1 \begin{bmatrix} \rho \\ \rho u_t \\ \rho u_n \\ \rho E \end{bmatrix} + N_2 \begin{bmatrix} U_1^2 \\ U_2^2 \\ U_3^2 \\ U_4^2 \end{bmatrix} + N_3 \begin{bmatrix} U_1^3 \\ U_2^3 \\ U_3^3 \\ U_4^3 \end{bmatrix}, \tag{A.9}$$

where $\mathbf{R}_1$, as defined in Eq. (A.4), is the rotation matrix which rotate the normal and tangential components for node 1 to the global Cartesian system. The same type of rotation must also take place on the test functions, i.e.,

$$\mathbf{W}_h = \sum_{i=1}^{4} N_1 \mathbf{R}_1 \mathbf{e}_i, \tag{A.10}$$

on the node 1.

After the coordinate rotation is set, as in Eq. (A.9) and (A.10), the element matrix is affected as follow

$$\mathbf{A}_{R}^{e} = \left[\begin{array}{c:c:c|c} \mathbf{R}_1^T \mathbf{A}_{11}^{e} \mathbf{R}_1 & \mathbf{R}_1^T \mathbf{A}_{12}^{e} & \mathbf{R}_1^T \mathbf{A}_{13}^{e} & \mathbf{R}_1^T \mathbf{A}_{14}^{e} \\ \hdashline \mathbf{A}_{21}^{e} \mathbf{R}_1 & & & \\ \hdashline \mathbf{A}_{31}^{e} \mathbf{R}_1 & & & \\ \hline \mathbf{A}_{41}^{e} \mathbf{R}_1 & & & \end{array}\right]. \tag{A.11}$$

Finally, once the solution is obtained, a post-processing is required in order to recover the components of the velocity on the global Cartesian system,

$$\mathbf{U}_{h}^{1} = \mathbf{R}_1^{-1} \mathbf{U}_{L}^{1}. \tag{A.12}$$