DÉVELOPPEMENT D'UNE MÉTHODE IMPLICITE SANS MATRICE POUR LA SIMULATION 2D-3D DES ÉCOULEMENTS COMPRESSIBLES ET FAIBLEMENT COMPRESSIBLES EN MAILLAGES NON-STRUCTURÉS

Thibaud Kloczko

N° : 2006 ENSAM 0007

ENSAM
Ecole Nationale Supérieure d'Arts et Métiers

École Doctorale n°432 : Science des Métiers de l'Ingénieur

THÈSE

pour obtenir le grade de

Docteur
de
l'École Nationale Supérieure d'Arts et Métiers

Spécialité "Mécanique"

présentée et soutenue publiquement
par

Thibaud KLOCZKO

le 15 mars 2006

DÉVELOPPEMENT D'UNE MÉTHODE IMPLICITE SANS MATRICE POUR LA SIMULATION 2D-3D DES ÉCOULEMENTS COMPRESSIBLES ET FAIBLEMENT COMPRESSIBLES EN MAILLAGES NON-STRUCTURÉS

Directeur de thèse : **Alain LERAT**
Codirecteur de thèse : **Christophe CORRE**

Jury :

Hervé GUILLARD, Directeur de recherche, SMASH, INRIA Sophia-Antipolis ..... Président
Erik DICK, Professeur, Université de Ghent (Belgique) ..... Rapporteur
Bruno DUBROCA, Professeur associé, MAB, Université Bordeaux I ..... Rapporteur
Alberto BECCANTINI, Ingénieur, LTMF, CEA Saclay ..... Examinateur
Christophe CORRE, Maître de Conférences, SINUMEF, ENSAM Paris ..... Examinateur
Alain LERAT, Professeur, SINUMEF, ENSAM Paris ..... Examinateur

Laboratoire de Simulation Numérique en Mécanique des Fluides
ENSAM, CER de Paris

À ma fiancée

et à ma famille...

Remerciements

Cette histoire commence en juillet 1999 lorsque, juste avant de soutenir l'oral de math du concours d'entrée à l'ENSAM, je suis passé devant la porte du laboratoire SINUMEF. Elle se poursuit quand à la fin de ma deuxième année, je fus admis en DEA par un certain professeur Lerat, et quand au moment de choisir le PFE avec mon ami Sylvain, nous avons opté pour le sujet le plus incompréhensible qui parlait de "sans matrice" et "d'implémentation". La suite constitue l'expérience la plus riche que j'ai vécue à ce jour : une thèse au laboratoire LTMF du CEA Saclay ; et je tiens à remercier les personnes avec qui j'ai partagé ces moments inoubliables.

Je suis très reconnaissant à Monsieur A. Lerat, Professeur à l'ENSAM Paris, pour avoir initié cette tranche de vie et pour avoir porté un sincère intérêt à mes travaux de recherche.

Je remercie vivement Messieurs E. Dick et B. Dubroca, respectivement Professeur à l'Université de Ghent (Belgique) et Professeur associé à l'Université Bordeaux I, pour avoir accepté de lire ce mémoire et porter un jugement sur mon travail.

Je tiens à remercier Monsieur H. Guillard, Directeur de recherche à l'INRIA Sophia-Antipolis, pour avoir accepté de participer au jury de thèse.

Parce que cette expérience n'aurait pas été la même sans eux, je souhaite remercier chaleureusement les membres du labo LTMF permanents et temporaires qui grâce à leurs (fortes?) personnalités contribuent à faire de la recherche un authentique plaisir. Je citerai en vrac : Sylvain (bien sûr), Isabelle, San-Ghir, Jérôme, Didier, Jean-Luc, M'sieur Métier, Hélène, Donna, Laure, Guillaume, Enrico, Nadia, Anne, Arnaud, Karima, Alex, Julie, Rattana...

Avec une mention spéciale pour : Sergueï ("Vive la Russie"), Stéphane ("Maître GMRES"), Etienne ("Banane d'argent"), Fred ("Banane d'or"), JPM ("Gzip! En voilà une commande qui déchire!"), Henri ("Merci Patron ;-) "), Véro ("Entrouve-moi!") et l'inoubliable Yong-Joon (dit "Choun-Choun")...

Le meilleur pour la fin. Les mots me manquent pour exprimer ce que je dois à Alberto Beccantini et à Christophe Corre. Ils me pardonneront certainement de ne pas citer leurs titres respectifs car, pour moi, plus que des encadrants, ce sont de véritables amis... Merci encore à vous deux et surtout à bientôt !!!

Contents

Introduction

Objectives and context of the work The present work is devoted to the development of an efficient implicit scheme for computing compressible and low-speed flows on unstructured meshes. In this introduction, we will first explain why the Commissariat à l'Énergie Atomique (CEA) is interested in the development of such an efficient flow solver for all-speed flows, which tools were available at the start of this thesis and what were their limitations; next we will detail the strategy followed to improve over the existing tools and enable the analysis of configurations that were previously out of range.

The global context of this work is the development by the CEA of numerical tools for studying by way of simulation the safety of nuclear plants or other energy-producing devices such as fuel cells and hydrogen energy. In this latter case for instance, a major issue is precisely to guarantee the use of hydrogen will remain safe when this highly energetic and strongly explosive gas will be made available to the general public. The European excellence network Hysafe (Safety of Hydrogen as an Energy Carrier) precisely aims at analyzing the risks of occurrence and possible consequences of a number of hydrogen-related accidental phenomena in order to establish hydrogen-technologies as a credible alternative to conventional sources of energy and to gain public acceptance for their daily use. In order to reach these goals, the members of the network plan to carry out numerous experimental and numerical studies; among these members, the Laboratoire d'Études des Transferts et de Mécanique des Fluides (LTMF / Laboratory for the study of transfer phenomena and of fluid mechanics), belonging to the CEA center located in Saclay, has developed a strong expertise on the study of the hydrogen risk in contained use through years of activity in the field of civil nuclear engineering and is now turning this expertise towards the field of fuel cells and hydrogen energy.

Let us now be a little more specific on the concept of hydrogen risk since it is a key factor in the development of a flow-solver for all-speed flows. At some stages of accident scenarios in nuclear plants, hydrogen may be released or generated inside a nuclear reactor such as the current and future REP (Réacteur à Eau Pressurisée) or PWR (Pressurised Water Reactor)[1]; a reactive mixture of air, water steam and hydrogen may then locally form and it is necessary to be able to accurately predict the distribution of these different gas species inside the geometry in order to correctly simulate the combustion process for hydrogen. Indeed, depending on the local concentration of species and on the presence of mitigation devices, hydrogen may burn following different modes (flame diffusion, local or global deflagration, accelerated flames, detonation) or not burn at all. So-called $0D$ models or Lumped-parameter models have been intensively used for years when carrying safety studies and have successively allowed to obtain averaged values for temperature, pressure and chemical species concentration. However, these models

[1] A similar situation may occur with hydrogen used as an energy carrier, in which case the nuclear reactor could be replaced with a common garage; this makes clear the natural application of the CEA expertise within the Hysafe network.

are unable to provide a detailed information on the local concentration of the gases or on a stratification process. Specific CFD tools have thus been developed in order to assess the risks of ignitability and detonation of gas mixtures by solving with high-resolution schemes the full 3D systems of conservation laws governing the flows under consideration : the LTMF has developed in particular the numerical platform CAST3M *Fluids* [4] as well as the TONUS code [6], in collaboration with the IRSN (Institut pour la Radioprotection et la Sûreté Nucléaire / National Institute for Radiation and Nuclear Safety).
The flows encountered when simulating hydrogen release and combustion in the large scale geometries typical of nuclear reactors are extremely varied, with a large spectrum of space and time characteristic scales and flow regimes : for instance, a stratification process may last for hours while a combustion process will be over in a matter of seconds; while the whole volume of a REP/PWR is about 50000 m^3, with geometrical obstacles measuring several meters, the typical length of a breach in the primary circuit will be of a few centimeters, which is also the size of the flame fronts that must be accurately computed in order to assess the induced dynamic loads; last, as already pointed out, hydrogen combustion may give rise to very weakly compressible flows for a slow deflagration or a flame diffusion but also to highly compressible phenomena when a denotation takes place. When confronted with such a wide spectrum of flows, two main strategies can be considered : the first one aims at developing separately the most appropriate and accurate numerical method for each type of flows while the second one is oriented towards the development of an "all-purpose" flow solver. Clearly, when developing the CAST3M platform, the necessity to be able to compute both (quasi) incompressible and compressible flows has been a major issue and the next paragraph of this introduction is entirely devoted to a quick description of the state of the art regarding the design of all-speed flow solvers, completed with an overview of the existing choices made within the CAST3M tool at the start of this work.

Development of all-speed flow solvers The "incompressible flows" and "compressible flows" scientific communities have long been two separate worlds, as illustrated by the fact reference books on the numerical methods specific to each of these flow regimes are usually almost exclusively devoted to one or the other (see for instance [27] for techniques specific to compressible flow problems such as the Godunov scheme and [24] for methods related to incompressible flow problems such as fractional step or projection methods). However, for some years now, a continuous effort has been devoted to building bridges between these two "worlds", motivated by the fact a number of real-world applications display simultaneously flow regions that remain essentially incompressible and others where compressibility effects occur and have to be taken into account; examples of interest for the CEA have been given above with the various phenomena involved in the hydrogen combustion process and another example related to aeronautical issues may be found in [62] with the study of a vertical take-off and landing aircraft where the flow over the complete aircraft in landing situation remains low-speed hence incompressible while the flow emitted from the manoeuvring jets is highly compressible.
As recalled in the recent review paper [42], the major difference between the incompressible and compressible Euler or Navier-Stokes equations lies in the continuity (or mass conservation) equation. Purely incompressible flow solvers rely on pressure-based methods, thus christened because the pressure is used as a mapping parameter to satisfy the continuity equation, which are usually applied in two steps : the momentum equations are solved in a first step, with a pressure gradient computed from the pressure in the previous time step or simply not taken into account, so as to yield an auxiliary velocity field; the pressure field is then computed in a second step, from a Poisson equation, in such a way the auxiliary velocity maps onto a divergence-free

velocity field, thus satisfying exactly the continuity equation. Purely compressible flow solvers are density-based, meaning they rely on the solution of the mass, momentum and energy equations written as an hyperbolic system of conservation laws for the set of conservative variables $(\rho, \rho\vec{V}, \rho E)$ where ρ is the density, $\vec{V}$ the velocity vector and E the specific total energy.
A first strategy to develop an all-speed flow solver could be therefore to implement both pressure-based and density based methods within the code. This was precisely done for the CAST3M software which provides on one hand a pressure-based solver, for computing incompressible or low-speed variable density flows such as encountered in stratification process and on the other hand a density-based solver using shock-capturing conservative techniques for computing detonation problems. More precisely, the pressure-based method is applied to the asymptotic approximation of the Navier-Stokes equations in the limit of small Mach numbers, which allows to filter out the acoustic waves; the space discretization is based on the Finite-Element method while the time advancement is performed through a semi-implicit projection algorithm. When dealing with detonation problems, the effects of viscosity are neglected so that the Euler equations remain to be solved using upwind discretization techniques within a cell-centered Finite-Volume formulation on general unstructured grids. The upwind schemes available in CAST3M are either of the flux vector splitting type (with Van Leer scheme [79] for instance), or of the flux difference splitting type, with the Roe scheme [67] or else hybrid schemes such as the AUSM+ scheme[46]. The time-integration is performed either explicitly or in an implicit way , using a Newton-Krylov approach.
A first drawback of a tool relying on such ad-hoc numerical techniques is the need to develop some a-priori knowledge on the flow of interest so as to be in position to select the proper technique. However, the main drawback of such a strategy is that it simply fails when the flows under study *simultaneously* display nearly incompressible and highly compressible flow regions. Indeed, asymptotic models are valid only when the Mach number is small enough (typically up to $M = 0.3$) while classical density-based methods fail to correctly compute inviscid flows in which the Mach number becomes too low (typically less than $M = 0.1$). Indeed, when the speed of the material wave becomes small with respect to the acoustic speed, the Euler system becomes stiff which leads to a very slow convergence rate to steady-state for standard density-based methods [75]; moreover, when $M \to 0$, some components of the numerical dissipation introduced by upwinding become excessively high, which yields poorly-accurate or even totally wrong flow solutions [28].

Turning to the second strategy, *i.e.* to the development of an all-speed or Mach-independent numerical method, requires either to extend the "incompressible" pressure-based methods to higher Mach number flows or to extend the "compressible" density-based methods to low Mach number flows. The first option has been explored since the beginning of the seventies [29] and more or less continuously since then [7], [68]; this line of work has probably culminated with the recent work [59]. The second option has been the object of thorough investigations since the mid-eighties, with the seminal work of Turkel [75], but finds its roots in one of the earliest attempt at drawing connections between incompressible and compressible flow solving techniques, namely the artificial compressibility method devised by Chorin [15]; this method turns the system of conservation laws governing an incompressible flow into an hyperbolic system of conservation laws that can be solved using one of the many schemes developed for the compressible Euler or Navier-Stokes equations (such as the Roe scheme for instance, provided the eigensystem can be computed). The application of the artificial compressibility method

remains limited to purely incompressible flows; however, Chorin's idea was later exploited by Turkel [75] who proposed to multiply the time-derivative of the Euler hyperbolic system with a so-called low-Mach preconditioning matrix designed to alter the acoustic speed associated with this system in such a way all eigenvalues of the flux Jacobian matrix remain of the same order when the Mach number goes to zero, thus avoiding the numerical stiffness issue. The preconditioning matrix introduced by Turkel was designed by working on a formulation of the Euler equations in terms of entropic variables $(p, \vec{V}, S)$, where p is the pressure and S the entropy, so as to decouple the pressure equation from the other evolution equations since only the pressure equation is to be affected by the application of preconditioning; simple matrix transformations allows then to recover the preconditioned Euler equations expressed in terms of conservative variables, to which numerical schemes developed for the fully compressible system can be directly applied. Following Turkel, several researchers such as Choi & Merkle [14], Weiss & Smith [85], Van Leer [80], Venkateswaran & Merkle [81], have proposed their own preconditioning matrix. Although derived from different philosophies and from different sets of variables (entropic for Turkel and Van Leer, viscous primitive $(p, \vec{V}, T)$ for Choi & Merkle and Weiss & Smith), all these preconditioners are closely related. For instance, they all become singular when $M = 0$ so they all require a cut-off value for their parameter. As already pointed out, preconditioning the system of conservation laws is not without effect on the numerical scheme applied to the system : for instance, when applying Roe's flux-difference splitting scheme to the preconditioned Euler equations, the matrix-based numerical dissipation must be constructed from the new eigensystem associated with this set of equations, fortunately available in closed form when the Turkel preconditioning is used (see [83]). Low-Mach number preconditioned versions of the AUSM+, Harten-Lax-van Leer or Jameson's CUSP schemes are respectively provided in [46] [51] [77].

The choice of an all-speed pressure-based solver or an all-speed density-based solver is essentially determined by the initial investment of the code developers towards one of this specifically targeted technique : in the case of the CAST3M code, the amount of efforts put into the development of high-resolution upwind schemes for solving compressible flows on general unstructured grids has motivated the implementation of Turkel low-Mach number preconditioning so as to give an all-speed capability to the solver. The preconditioned upwind schemes (Roe, AUSM+) implemented within CAST3M have proved their ability to yield accurate solutions of both weakly and highly compressible flows, with a convergence to steady state largely independent from the freestream Mach number, all properties precisely expected from the use of a low-Mach preconditioning [5]. In practice, it was observed however that schemes including preconditioning may display some lack of robustness (*i.e.* the computation blows up because of the occurrence of non-physical states, such as negative values of the pressure, within the flow); this robustness issue seems to be related with the use and definition of a cut-off (or limiting) value prescribed when computing the parameters appearing in the preconditioning matrix in order to prevent the matrix from becoming singular at $M = 0$, namely for stagnation flow. Although this "cut-off" definition is now used in every computation, it is still difficult to optimize its value for complex problem. Several definitions of the "cut-off" are available, which depend on the nature of the flow (inviscid or viscous, steady or unsteady) and also on the grid used for the computation (regular or stretched, which may introduce an aspect ratio in the cut-off definition). Moreover, it is possible to use a global "cut-off" for the whole computational domain or a local "cut-off" in each control cell (*cf.* [76] for a large review of such cases). Some authors tried to unify all the definitions by distinguishing each case [81]; nevertheless, it appears to be still challenging to get a simple but efficient "cut-off" definition. This point will be dealt with in this work but was not

the key issue that motivated the present study. In fact, once solved the all-speed flow issue and the solver applied to real-life computations involving a very large number of grid points because of the need to discretize simultaneously the large-scale geometry of a containment and specific regions such as the neighborhood of a breach, it appeared the Newton-Krylov implicit treatment used for speeding-up the convergence to steady-state was too memory-consuming. The first motivation of this work was therefore to develop, implement and validate within CAST3M an alternative implicit treatment which could offer a reduced memory requirements while preserving the global efficiency of the code, with the additional and crucial constraint to provide such properties for the whole range of Mach numbers, that is including the use of preconditioning. The next paragraph will now describe the strategy followed in this work to improve over the existing implicit scheme in the specific context of a preconditioned compressible flow solver.

Low-cost implicit treatments for all-speed flows Since the first objective of this work is to design a low cost implicit scheme for all-speed flows, where low-cost means both low computational cost and low storage, a first step is naturally to examine the existing low or at-least reduced cost strategies for compressible flow solvers; the second step will be the adaptation of such strategies to preconditioned schemes. Designing a low-cost implicit treatment means solving a multi-objective optimization problem in which one looks for the simultaneous minimization of the computational time involved in the convergence to steady (physical or dual) state and of the associated storage requirement. The minimization of computational cost can be performed by considering a two-objective problem since the global computational cost spent to reach a steady-state is a combination of intrinsic efficiency (*i.e.* the number of iterations needed to reach this steady-state) and unit cost per iteration. Overall, designing an efficient or low-cost implicit scheme means finding a solution of the 3-objective optimization problem summarized in Fig. 0.0.1. Usually, the maximization of intrinsic efficiency, that is the minimization of the

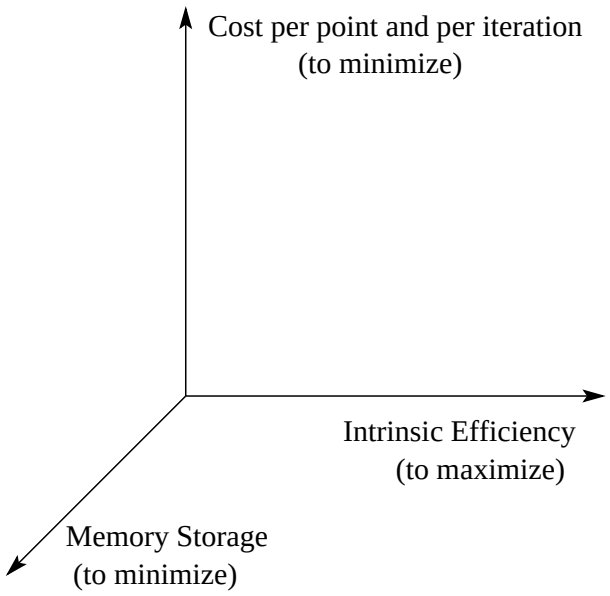


Figure 0.0.1: Design of an efficient implicit treatment : a multi-objective optimization problem.

number of iterations needed to achieve steady-state, and the minimization of the cost per point per iteration are conflicting objectives so that there is no unique solution to the multi-objective optimization problem but a set of optimal trade-off solutions, known as the Pareto optimal set. Thus, the Newton-Krylov method available in CAST3M may be considered as one of these optimal choices, which minimizes the number of iterations needed to reach steady-state, hence provides maximal intrinsic efficiency, at the expense of a rather high unit cost; moreover, the Newton-Krylov method is definitely not optimal as far as the storage objective is concerned. There exists a number of other optimal choices with respect to the two objectives related with the computational cost and we wish to select among those choices the one(s) yielding the lower memory requirements. The excessive memory demand introduced by an implicit stage can be often related to the use of a *block* implicit stage, based on flux Jacobian matrices introduced when linearizing the numerical fluxes in order to derive the implicit part of the scheme; moreover the unit cost of a solution method for a block implicit stage is usually rather high. However, the main advantage of using a block implicit stage comes from its high intrinsic efficiency, which explains block approaches may still constitute optimal choices as far as computational cost is concerned. A way to reduce memory requirement for an implicit method is to develop implicit treatments that do not rely on full Jacobian matrices, which means some kind of simplification has to be introduced in the explicit stage linearization; in turn, such a simplification is likely to induce a loss of intrinsic efficiency which will have to be compensated through a lower unit cost, so as to preserve the global efficiency. Following this line of idea, Pulliam et Chaussée proposed in 1981 a diagonalized implicit treatment applied to alternate direction factorization method [65] : using diagonal matrices for each $1D$ implicit stage instead of full matrices allows to reduce both the unit cost and memory requirement of the original implicit stage, even though some approximations are made when deriving this diagonal implicit stage, which leads to a slightly lower intrinsic efficiency with respect to the baseline block treatment. More recently, Buelow *et al.* have extended this diagonal approximate factorization technique to the case of preconditioned schemes within a dual-time framework and have successfully computed unsteady low-Mach number flows [10]; this same technique has been implemented by Pandya *et al.* within the NASA code OVERFLOW [62]. In the mid-nineties, Corre and Lerat have applied similar diagonalization ideas to line-relaxation methods of Jacobi and Gauss-Seidel type [21] and, in 2004, Corre and Kloczko have derived an extension of these ideas to preconditioned schemes [20]. It was estimated, however, that such diagonal or quasi-diagonal treatments, even though they allow a substantial reduction of memory requirements with respect to more standard block treatments while preserving an equivalent level of global computational cost (loss of intrinsic efficiency balanced by unit cost reduction), still remain too-memory consuming for grids containing very large number of grids points, likely to be used in the applications targeted at CEA. Moreover, when computing flows on structured grids, the diagonalization is successively performed along each grid direction; the extension of these ideas to unstructured grids means the diagonalization process will have to be successively applied on each face of a control volume, in the direction normal to that face, which may become quite expensive. It was therefore decided to opt for even more simplified methods such as matrix-free implicit methods. A well-known prototype of such methods is the implicit residual smoothing technique initially introduced in [45] for the Lax-Wendroff scheme and made popular by Jameson [35] : the basic idea of the residual smoothing technique applied to the Lax-Wendroff implicit scheme is to replace the square of the fluxes Jacobian matrices appearing in the implicit stage with their respective spectral radius (an operation made possible without altering the unconditional linear stability of the scheme thanks to the definite positive character of these dissipation matrices). The linear system associated with this simplified implicit stage is

no longer a block system but is purely scalar : it can then be solved for a reduced computational cost per iteration and with very low memory requirements; naturally, the price to pay for this simplification from block to scalar is a loss of intrinsic efficiency. However, it was found in practice that this loss of intrinsic efficiency was usually balanced by a truly low unit cost, resulting eventually in a global efficiency as good as that of the original block treatment, with the benefit of a reduced memory storage. Shortly after the introduction of this spectral radius simplification in [45] Jameson and Baker demonstrated in [35] the same matrix-free implicit stage could be successively coupled with the explicit Jameson's scheme [37] based on a centered discretization completed with a well-tuned scalar artificial viscosity : in spite of the lack of consistency between the explicit and implicit stage, the resulting scheme had good stability properties and was much more efficient than a purely explicit scheme for a modest extra-cost due to the very simple form of this matrix-free implicit stage. Similar ideas made their way in the context of upwind schemes and though it seems difficult to trace back very precisely the origin of matrix-free implicit upwind schemes, a first contribution can be found in the work of Jameson and Yoon [38] where the absolute values of Jacobian matrices appearing in a block implicit stage are replaced with their respective spectral radius, taking advantage here again of the definite positive character of these dissipation matrices. However, contrary to the case of the Lax-Wendroff scheme where this spectral radius simplification is sufficient to make the implicit stage matrix-free, the implicit stage associated with (typically) Roe upwind scheme also contains flux Jacobian matrices that cannot be simplified since they are not definite positive in general; however, since these flux Jacobian matrices appear in the implicit stage as multiplied by time-increments of the conserved variables, it is possible to replace such products with time-increments of the flux-vectors and to relax these new unknown when solving the resulting matrix-free implicit stage. It seems the idea of this approximation, which allows to turn a block implicit upwind scheme into a matrix-free implicit scheme originates in the work of Sharov and Nakahashi [71]. Using simultaneously the spectral radius simplification and the flux-vector increment relaxation, Löhner *et al.* have successfully developed a matrix-free implicit method for solving the 3D Navier-Stokes equations on unstructured grids [48], which has been next further extended to unsteady flows [49] as well as to all-speed flows using a preconditioning technique [50]. Let us emphasize once again the intrinsic efficiency provided by such a matrix-free implicit stage is rather low : however its extremely low cost per iteration makes it globally as efficient as (or even more efficient than) the original block implicit stage; moreover, its modest memory requirements make it truly attractive for large scale computations. It was then decided at the start of this thesis to focus our work on the development of a matrix-free implicit scheme such as initially introduced in [48] as a low-cost alternative to the existing implicit treatment available in CAST3M, with the clear objective of making large-scale computations accessible with this improved version. Since the intrinsic efficiency of matrix-free methods is quite poor, it is absolutely crucial to make the unit cost of these methods as low as possible in order to preserve their global efficiency. A difficulty appears when a low-Mach preconditioning matrix is introduced for viscous or unsteady flow computations since in that case it is no longer possible to make the implicit stage truly matrix-free; it will be shown however in this work that a proper formulation of the implicit stage, which takes advantage of properties specific to Turkel preconditioning matrix, allows to preserve an extremely low cost per iteration for all-speed viscous unsteady flows, thus making the proposed matrix-free approach a viable implicit strategy to be used within CAST3M.

Research work organization and presentation The research work has been organized in three main stages, around a numerical tool associated with each stage :

- in the first stage the intrinsic efficiency of the proposed matrix-free treatment has been studied following a Von Neumann analysis; a code has been developed in order to compute the amplification matrix associated with this scheme for the linearized Euler and Navier-Stokes equations in the steady and unsteady case (this latter being treated within a dual time framework) and to compare the matrix-free treatment, from the viewpoint of intrinsic efficiency, with block treatments, for compressible as well as low-Mach number flows. Von Neumann analysis has also been used to assess the intrinsic performances of various implicit treatments (Jacobi relaxation, Lower-Upper Successive Gauss-Seidel) that can be used to solve the matrix-free implicit stage.

- in the second stage, some of the most interesting implicit treatments selected on the basis of the previous study have been implemented in a specially-developed structured code in order to assess their global performance as well as their associated memory requirements. This intermediate step, prior to any implementation within the general-purpose CAST3M code, has allowed to evaluate efficiently and with great flexibility the numerous choices that have been investigated throughout this work (be it implicit strategies or cut-off choices for low-Mach preconditioning for instance). This evaluation has been systematically performed on a set of well-chosen test-problems and the most efficient treatments selected on the basis of this second stage have been actually implemented within CAST3M.

- in the last stage of our work, the matrix-free treatment previously developed in the context of finite-difference schemes on structured grids has been extended to a finite-volume formulation for unstructured grid computations and implemented within CAST3M. A point-Jacobi and LU-SGS solution of the matrix-free implicit stage have been developed and these new implicit treatments have been compared with the already available block implicit stage solved with a Newton-Krylov approach both for weakly compressible and compressible, steady and unsteady flows, demonstrating the superior efficiency of the newly implemented matrix-free treatment. The preconditioned compressible flow solver has also been successfully compared with the existing pressure-based solver applied to an asymptotic model.

The detailed presentation of our research work in this report follows the above chronology :

- the first part of the report is devoted to the review and analysis of some standard block-implicit treatments for the two-dimensional Euler and Navier-Stokes equations with a view to identify the best candidate for a fair comparison with the matrix-free treatment introduced in the second part of the report. A 2D framework is retained for the sake of presentation simplicity only : the $2D$ analysis extends immediately to $3D$ problems and all the later developments have been carried out in $3D$. Chapter 1 described the design principles for building a block implicit upwind scheme that can be applied for all regimes of inviscid or viscous, steady or unsteady flows, using low-Mach preconditioning and dual-time stepping. Chapter 2 details various techniques (approximation factorization in its basic or modified form, point or line-relaxation) for solving the block linear system associated with the previous scheme. These first two chapters also allow to introduce the notations (discretization operators in particular) that will be used throughout this report. In chapter 3, the linear stability and intrinsic efficiency of the previously presented implicit treatments is studied using a Von Neumann analysis and the hierarchy deduced from this analysis allows to select an alternate line-relaxation treatment as the reference efficient (but

memory-consuming) treatment to which the proposed matrix-free method will be compared to.

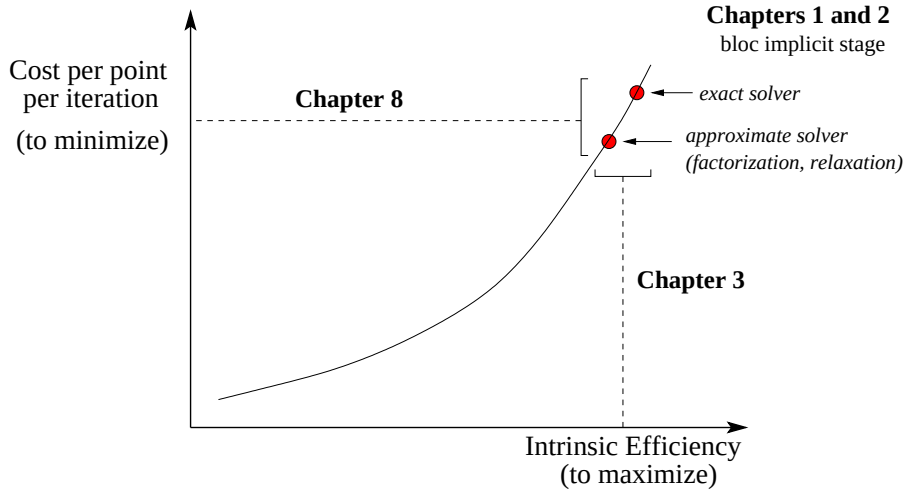


Figure 0.0.2: Road map for the design of an efficient implicit treatment : cost-analysis of standard block-implicit treatments.

- the second part of the report forms the main original contribution of this research work, namely the description and systematic analysis of a matrix-free treatment that can be applied to any type of flow (inviscid/viscous, low-Mach/highly compressible, steady/unsteady). Chapter 4 describe the design principles of a matrix-free method for solving the steady Euler equations as well as the extensions to perform when solving the Navier-Stokes equations and unsteady flows within a dual time-step approach. Taking advantage of the properties of Turkel preconditioning matrix, an optimized implicit treatment is introduced, that allows to preserve an extremely low unit cost even when the low-Mach preconditioning is active. Chapters 5,6 and 7 are then devoted to the Von Neumann analysis of the matrix-free treatment, when applied respectively to the steady (chapters 5 and 6) and unsteady (chapter 7) Euler and Navier-Stokes equations, with a particular emphasis on low-Mach number flows. The loss of intrinsic efficiency introduced by the matrix-free simplifications is thus clearly evidenced, which makes crucial the requirement of an extremely low unit cost to preserve global efficiency - this reduced cost being ensured by the proposed matrix-free treatment. The flexibility offered by the Von Neumann analysis also allows to analyze the modifications to apply to the standard choice of low-Mach preconditioning cut-off when computing viscous or unsteady flows. Finally, chapter 8 presents the results obtained for some significant test-cases computed using either a high-intrinsic-efficiency block-treatment or the proposed low-cost matrix-free treatment. The advantages brought by this latter approach, both in terms of global computational time and memory requirement, are clearly demonstrated and motivate the implementation of the matrix-free treatment within the CAST3M numerical platform, as described in the last part of the report.

- the third part of the report deals with the implementation of the proposed matrix-free treatment within CAST3M and the demonstration of its advantages over existing tech-

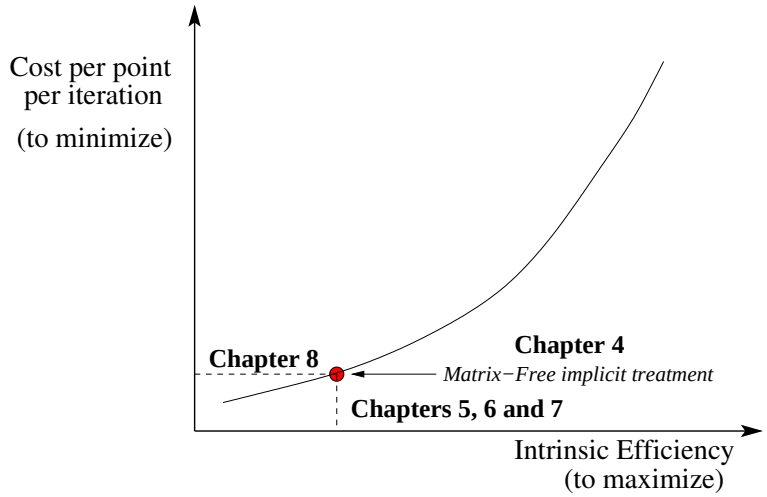


Figure 0.0.3: Road map for the design of an efficient implicit treatment : cost-analysis of the matrix-free treatment.

niques for computing applications of interest for the CEA. Chapter 9 is thus devoted to the extension of the matrix-free approach to a finite-volume formulation on unstructured grids. It is shown the high simplicity of the method is preserved without particular difficulty. Chapter 10 presents some comparisons between the Newton-Krylov implicit method available in CAST3M and the newly implemented approach : a classical 2D inviscid flow over a bump is first computed to demonstrate the computational time reduction offered by the matrix-free treatment; the same problem is then computed in 3D to make even clearer the gain obtained as far as the memory storage is concerned. The 11^{th} and last chapter displays a comparison between the asymptotic solver available in CAST3M and the newly implemented low-Mach matrix-free treatment for the case of laminar, low-Mach number steady and unsteady flows in a Tee junction. The results provided by the matrix-free treatment confirm its interest for the efficient computation of low-Mach number (single-species) flows.

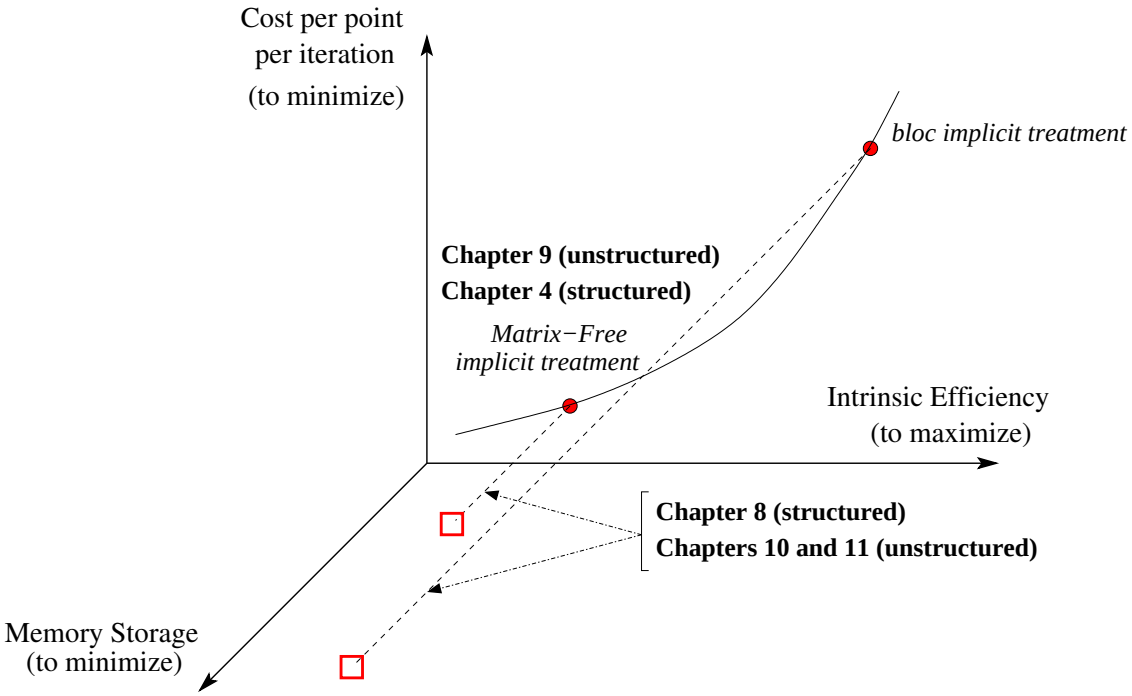


Figure 0.0.4: Road map for the design of an efficient implicit treatment : global (cost and memory requirement) analysis.

Part I

Analysis of Classical Implicit Methods

Chapter 1

Classical Implicit Upwind Scheme

This first chapter provides the basics of the physical modeling and of the numerical methods used throughout this study : all the flow problems considered in this work can be described using either the Euler or the Navier-Stokes equations written for a perfect gas, which are briefly recalled in a first section. These equations are then approximated with an upwind scheme which is made implicit in order to increase the scheme's convergence to steady state. In the first part of the study, implicit schemes will be analyzed on Cartesian grids so that the general principles governing the design of such conventional block implicit schemes are recalled within a finite difference framework in section 2. Some usual simplifications when designing implicit stages coupled with high order explicit stages for viscous flow problems are also pointed out. Since our work is focused on the design of an efficient implicit scheme for all-speed flows, the use of a preconditioning strategy when computing low-Mach number problems and its impact on the space and time discretization are detailed in section 3. The last section is devoted to a brief reminder of the dual-time technique used in the present study for computing low-Mach number unsteady flows : this technique is retained because it will allow to take advantage, when dealing with unsteady flows, of the developments performed for reaching steady-state at a reduced computational cost for steady flows.

1.1 Governing Equations

1.1.1 Navier-Stokes equations for a perfect gas flow

The Navier-Stokes equations express the conservation of mass, momentum and energy for a compressible viscous fluid flow. When written using dimensional quantities, denoted by superscript $\tilde{}$ hereafter, the differential conservative form of these conservation laws reads :

$$\frac{\partial \tilde{\rho}}{\partial \tilde{t}} + \vec{\tilde{\nabla}} \cdot (\tilde{\rho}\vec{\tilde{u}}) = 0 \ , \tag{1.1.1a}$$

$$\frac{\partial \tilde{\rho}\vec{\tilde{u}}}{\partial \tilde{t}} + \vec{\tilde{\nabla}} \cdot (\tilde{\rho}\vec{\tilde{u}}\ \vec{\tilde{u}}) + \vec{\tilde{\nabla}}\tilde{p} = \vec{\tilde{\nabla}} \cdot \tilde{\underline{\tau}} + \tilde{\rho}\vec{\tilde{g}} \ , \tag{1.1.1b}$$

$$\frac{\partial \tilde{\rho}\tilde{E}}{\partial \tilde{t}} + \vec{\tilde{\nabla}} \cdot (\tilde{\rho}\vec{\tilde{u}}\tilde{H}) = \vec{\tilde{\nabla}} \cdot (\tilde{\lambda}\vec{\tilde{\nabla}}\tilde{T}) + \vec{\tilde{\nabla}} \cdot (\tilde{\underline{\tau}} \cdot \vec{\tilde{u}}) + \tilde{\rho}\vec{\tilde{g}} \cdot \vec{\tilde{u}} \ . \tag{1.1.1c}$$

where $\tilde{\rho}$ represents the gas density, $\vec{\tilde{u}}$ is the velocity vector, $\tilde{p}$ is the static pressure, and $\tilde{E}$ and $\tilde{H}$ are the specific total energy and specific total enthalpy, related by the following equation,

$$\tilde{H} \;=\; \tilde{E} \;+\; \frac{\tilde{p}}{\tilde{\rho}} \;. \tag{1.1.2}$$

In the right-hand-side (RHS) of (1.1.1b), (1.1.1c), $\tilde{T}$ is the temperature, $\tilde{\lambda}$ the thermal conductivity, $\vec{\tilde{g}}$ the gravitational acceleration and $\underline{\tilde{\tau}}$ denotes the shear stress tensor which, for a Newtonian fluid, is given by :

$$\underline{\tilde{\tau}} \;=\; \tilde{\mu}\left(\vec{\tilde{\nabla}}\vec{\tilde{u}} + (\vec{\tilde{\nabla}}\vec{\tilde{u}})^T\right) - \;\frac{2}{3}\;\tilde{\mu}(\vec{\tilde{\nabla}}\cdot\vec{\tilde{u}})\underline{I} \tag{1.1.3}$$

where $\tilde{\mu}$ is the dynamic viscosity (when the viscous effects are negligible, one can take $\tilde{\mu} = 0$ and one recovers the hyperbolic system of the Euler equations).

The system is closed by the equation of state for a single-species calorically perfect gas,

$$\tilde{p} \;=\; \tilde{\rho}\tilde{R}\tilde{T} \;=\; (\gamma-1)\tilde{\rho}\left(\tilde{E} - \frac{1}{2}|\vec{\tilde{u}}|^2\right) \;=\; (\gamma-1)\tilde{\rho}\tilde{e} \tag{1.1.4}$$

where $\tilde{e}$ is the specific internal energy, $\tilde{R}$ is the specific gas constant and γ is the ratio of specific heats, $\gamma = \frac{C_p}{C_v}$, equal to 1.4 for air at moderate temperatures. Eventually, the sound speed is given by :

$$\tilde{c}^{\,2} = \gamma\;\tilde{R}\;\tilde{T} = \frac{\gamma\;\tilde{p}}{\tilde{\rho}} \tag{1.1.5}$$

1.1.2 Non-dimensional form

The equations (1.1.1a) to (1.1.4) are made non-dimensional by using reference quantities denoted by the subscript ∞, *e.g.* free stream conditions, and a length scale $\tilde{L}$ specific to the considered flow problem (the channel height for internal flows for instance). Non-dimensional quantities are then defined by :

$$\vec{r} = \frac{\vec{\tilde{r}}}{\tilde{L}} \quad , \quad \rho = \frac{\tilde{\rho}}{\tilde{\rho}_\infty} \quad , \quad T = \frac{\tilde{T}}{\tilde{T}_\infty} \quad , \quad \mu = \frac{\tilde{\mu}}{\tilde{\mu}_\infty} \quad , \quad \lambda = \frac{\tilde{\lambda}}{\tilde{\lambda}_\infty} \quad , \quad \vec{e}_g = \frac{\vec{\tilde{g}}}{\tilde{g}} \quad , \tag{1.1.6}$$

and this first set of reference values can be used to define next a reference velocity, pressure, time and energy as follows :

$$\vec{u} = \frac{\vec{\tilde{u}}}{\sqrt{\tilde{R}\tilde{T}_\infty}} = \frac{\vec{\tilde{u}}}{\tilde{u}_\infty} \quad , \quad p = \frac{\tilde{p}}{\tilde{\rho}_\infty \tilde{u}_\infty^2} \quad , \quad t = \frac{t}{\tilde{L}/\tilde{u}_\infty} \quad , \quad E = \frac{\tilde{E}}{\tilde{u}_\infty^2} \quad , \quad H = \frac{\tilde{H}}{\tilde{u}_\infty^2} \quad . \tag{1.1.7}$$

Inserting these non-dimensional quantities into the original Navier-Stokes equations yields the non-dimensional form of the system :

$$\frac{\partial \rho}{\partial t} + \vec{\nabla}\cdot(\rho\vec{u}) = 0 \;, \tag{1.1.8a}$$

$$\frac{\partial \rho\vec{u}}{\partial t} + \vec{\nabla}\cdot(\rho\vec{u}\;\vec{u}) + \vec{\nabla}p = \frac{1}{Re}\;(\vec{\nabla}\cdot\underline{\tau}) + \frac{1}{Fr}\;\rho\vec{e_g} \;, \tag{1.1.8b}$$

$$\frac{\partial \rho E}{\partial t} + \vec{\nabla}\cdot(\rho\vec{u}H) = \frac{\gamma}{(\gamma-1)\;\;Pr\;Re}\;\vec{\nabla}\cdot(\mu\vec{\nabla}T) + \frac{1}{Fr}\;\rho\vec{e_g}\cdot\vec{u} + \frac{1}{Re}\;\vec{\nabla}\cdot(\underline{\tau}\cdot\vec{u}) \;, \tag{1.1.8c}$$

where standard non-dimensional numbers such as the Reynolds, Froude and Prandtl numbers have been introduced :

$$Re = \frac{\tilde{\rho}_\infty \tilde{u}_\infty \tilde{L}}{\tilde{\mu}_\infty} \quad , \qquad Fr = \frac{\tilde{u}_\infty^2}{\tilde{g}\tilde{L}} \quad , \qquad Pr = \frac{\tilde{C}_p \tilde{\mu}_\infty}{\tilde{\lambda}_\infty} \quad . \tag{1.1.9}$$

The non-dimensional form of the equation of state reads :

$$p = \rho T = (\gamma - 1)\rho \left(E - \frac{1}{2}|\vec{u}|^2 \right) = (\gamma - 1)\rho e \quad . \tag{1.1.10}$$

and the dimensionless sound speed is :

$$c^2 = \gamma \cdot T = \frac{\gamma \cdot p}{\rho} \tag{1.1.11}$$

In the remainder of this chapter, only the 2D version of the Navier-Stokes equations without the gravity source term will be considered for the sake of presentation simplicity. Note however the methods here described in 2D are readily extended to 3D problems : such an extension has been actually performed and the chapter devoted to the application of the numerical method developed in this work will naturally present some 3D test-cases. The two-dimensional Navier-Stokes equations in the absence of gravity effects can be put in the following conservative form :

$$w_t + (f^E(w) - f^V(w, w_x, w_y))_x + (g^E(w) - g^V(w, w_x, w_y))_y = 0 \tag{1.1.12}$$

where t is the time, x and y are the space coordinates, $w = w(x, y, t)$ is the vector of conserved variables, f^E and g^E are the inviscid flux vector functions and f^V and g^V are the viscous flux vector functions :

$$\begin{aligned} w &= \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} \quad , \quad f^E = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix} \quad , \quad g^E = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{pmatrix} \quad , \\ f^V &= \frac{1}{Re} \begin{pmatrix} 0 \\ \tau_{xx} = \frac{4}{3}\mu u_x - \frac{2}{3}\mu v_y \\ \tau_{xy} = \mu(u_y + v_x) \\ u\tau_{xx} + v\tau_{xy} + \frac{\gamma\mu}{Pr} e_x \end{pmatrix} \quad , \quad g^V = \frac{1}{Re} \begin{pmatrix} 0 \\ \tau_{yx} = \mu(u_y + v_x) \\ \tau_{yy} = \frac{4}{3}\mu v_y - \frac{2}{3}\mu u_x \\ u\tau_{yx} + v\tau_{yy} + \frac{\gamma\mu}{Pr} e_y \end{pmatrix} \quad . \end{aligned} \tag{1.1.13}$$

Let us also introduce the Jacobian matrices of the inviscid and viscous flux functions that will be needed later on, when deriving an implicit scheme for solving system (1.1.12) :

$$\begin{aligned} A^E(w) &= \frac{df^E}{dw}(w) \quad , & B^E(w) &= \frac{dg^E}{dw}(w) \quad , \\ A_0^V(w, w_x, w_y) &= \frac{\partial f^V}{\partial w}(w, w_x, w_y) \quad , & B_0^V(w, w_x, w_y) &= \frac{\partial g^V}{\partial w}(w, w_x, w_y) \quad , \\ A_1^V(w, w_x, w_y) &= \frac{\partial f^V}{\partial w_x}(w, w_x, w_y) \quad , & B_1^V(w, w_x, w_y) &= \frac{\partial g^V}{\partial w_x}(w, w_x, w_y) \quad , \\ A_2^V(w, w_x, w_y) &= \frac{\partial f^V}{\partial w_y}(w, w_x, w_y) \quad , & B_2^V(w, w_x, w_y) &= \frac{\partial g^V}{\partial w_y}(w, w_x, w_y) \quad . \end{aligned} \tag{1.1.14}$$

The (well-known) expressions of these matrices are summarized in Appendix A.

1.2 Standard Block Implicit Scheme

Let us now recall the standard steps to build a block implicit scheme for solving the Navier-Stokes equations. Since our interest will be restricted to discretization schemes that treat separately the convective and diffusive physical fluxes and since, moreover, the viscous fluxes discretization is straightforward because purely centered, we will first consider the derivation of a block implicit upwind scheme for the Euler equations before proceeding to the Navier-Stokes case.

1.2.1 Euler implicit scheme

The two-dimensional Euler equations written in conservative form read :

$$w_t \; + \; f^E(w)_x \; + \; g^E(w)_y \; = \; 0 \tag{1.2.1}$$

Let $\phi_{i,j}$ be a mesh function defined on a uniform Cartesian grid ($x_i = i\delta x$, $y_j = j\delta y$), with constant steps δx and δy and let us introduce the following basic difference and average operators acting in each grid direction :

$$\begin{aligned} &(\delta_1\phi)_{i+\frac{1}{2},j} = \phi_{i+1,j} - \phi_{i,j} \quad , \qquad (\delta_2\phi)_{i,j+\frac{1}{2}} = \phi_{i,j+1} - \phi_{i,j} \quad , \\ &(\mu_1\phi)_{i+\frac{1}{2},j} = \tfrac{1}{2}(\phi_{i+1,j} + \phi_{i,j}) \quad , \quad (\mu_2\phi)_{i,j+\frac{1}{2}} = \tfrac{1}{2}(\phi_{i,j+1} + \phi_{i,j}) \quad . \end{aligned} \tag{1.2.2}$$

The time step used for time integration will be denoted Δt and the ratio between this time step and the space step in each direction writes : $\sigma_1 = \dfrac{\Delta t}{\delta x}$, $\sigma_2 = \frac{\Delta t}{\delta y}$.

Explicit stage

The exact solution of system (1.2.1) can be approximated by an explicit scheme formally written as follows :

$$\frac{w_{i,j}^{n+1} - w_{i,j}^n}{\Delta t} + \mathcal{E}(w^n) = 0 \tag{1.2.3}$$

where $w_{i,j}^n$ denotes the numerical solution at time $n\Delta t$ and mesh point ($x = i\delta x$, $y = j\delta y$), $\mathcal{E}$ is the space operator defining the discretization scheme. In the case of a conservative scheme, operator $\mathcal{E}$ can be expressed in the form :

$$\mathcal{E}(w^n) = \left(\frac{\delta_1 F}{\delta x} + \frac{\delta_2 G}{\delta y}\right)_{i,j}^n = \frac{1}{\delta x \delta y}\left((F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j})\;\delta x \; + \; (G_{i,j+\frac{1}{2}} - G_{i,j-\frac{1}{2}})\;\delta y\right)^n \tag{1.2.4}$$

where the equivalence between a conservative finite-difference treatment and a finite-volume formulation on a Cartesian grid has been emphasized. The quantities F and G are the numerical fluxes approximating the inviscid fluxes f^E and g^E at the faces of the control cell (i,j) (*cf.* figure 1.2.1). In the most simple approach, these numerical fluxes can be simply evaluated

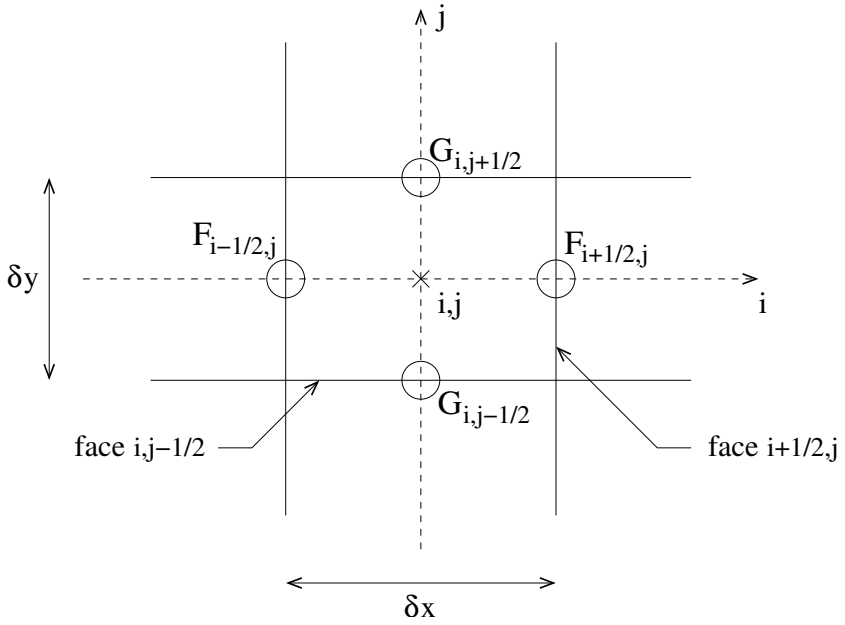


Figure 1.2.1: Control cell used to write the scheme

at each face by using the physical states in the cells located on each side of a given face (*cf.* figure 1.2.2) :

$$\begin{aligned} F_{i+\frac{1}{2},j} &= F(w^L_{i+\frac{1}{2},j}, w^R_{i+\frac{1}{2},j}) \qquad \text{with} \qquad w^L_{i+\frac{1}{2},j} = w_{i,j} \quad , \quad w^R_{i+\frac{1}{2},j} = w_{i+1,j} \quad , \\ G_{i,j-\frac{1}{2}} &= G(w^L_{i,j-\frac{1}{2}}, w^R_{i,j-\frac{1}{2}}) \qquad \text{with} \qquad w^L_{i,j-\frac{1}{2}} = w_{i,j-1} \quad , \quad w^R_{i,j-\frac{1}{2}} = w_{i,j} \quad . \end{aligned} \tag{1.2.5}$$

For a large class of schemes, these numerical fluxes can be cast into the form :

$$\begin{aligned} F^n_{i+\frac{1}{2},j} &= ((\mu_1 f^E)^n - d_1(w^n))_{i+\frac{1}{2},j} \\ G^n_{i,j-\frac{1}{2}} &= ((\mu_2 g^E)^n - d_2(w^n))_{i,j-\frac{1}{2}} \end{aligned} \tag{1.2.6}$$

where $\mu_1 f^E$, $\mu_2 g^E$ correspond to a simply centered approximations of the inviscid fluxes, while $d_1(w^n)$, $d_2(w^n)$ denote the numerical dissipation fluxes. Many choices of numerical dissipation are available in the literature but they can be roughly categorized into scalar or matrix dissipation [74]. Thus the numerical dissipation fluxes take eventually the form :

$$d_1(w) = D_1(w)\delta_1 w \quad , \quad d_2(w) = D_2(w)\delta_2 w \quad , \tag{1.2.7}$$

where $D_1(w)$ and $D_2(w)$ are the numerical dissipation (either scalar or matrix) coefficients. In the case of the Roe scheme [67], these coefficients are matrices defined by :

$$D_1(w) = \tfrac{1}{2}|A^E_R| \quad , \quad D_2(w) = \tfrac{1}{2}|B^E_R| \quad , \tag{1.2.8}$$

where A^E_R (resp. B^E_R) denotes the Roe average of the Jacobian A^E (resp. B^E). For the Rusanov scheme [69], the coefficients D_i are scalars :

$$D_1(w) = \tfrac{1}{2}\rho(A^E) \quad , \quad D_2(w) = \tfrac{1}{2}\rho(B^E) \quad , \tag{1.2.9}$$

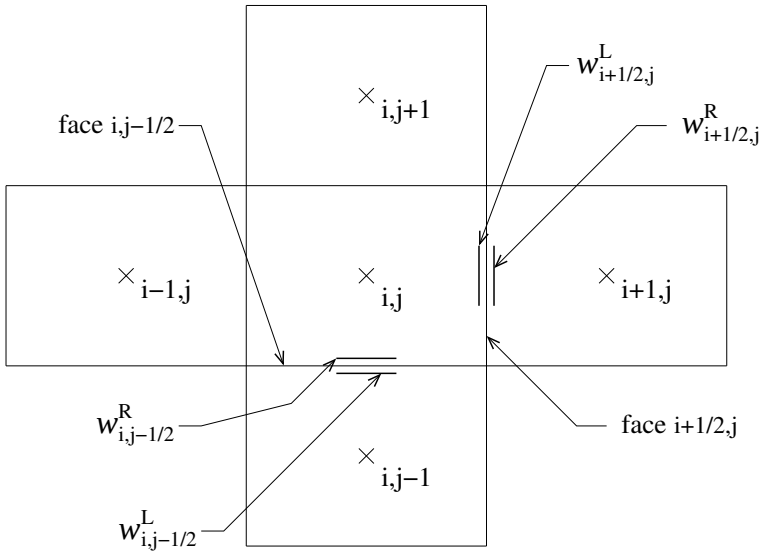


Figure 1.2.2: Classical stencil for a first-order space accurate upwind scheme

where $\rho(A^E)$ (resp. $\rho(B^E)$) denotes the spectral radius of the Jacobian A^E (resp. B^E).

Implicit stage

The explicit scheme (1.2.3) is submitted to a restrictive so-called CFL condition on the time-step that can be used in the convergence process to a steady-state. An implicit version of the scheme can be developed which allows to get rid of any stability restriction on the choice of Δt (for linear problems at least). Making the scheme implicit with respect to time t comes to replace the numerical flux F^n (resp. G^n) with its evaluation F^{n+1} (resp. G^{n+1}) at the new time-level, leading to :

$$\frac{w_{i,j}^{n+1} - w_{i,j}^n}{\Delta t} + \left(\frac{\delta_1 F}{\delta x} + \frac{\delta_2 G}{\delta y} \right)_{i,j}^{n+1} = 0 \tag{1.2.10}$$

Introducing the increment $\Delta w^n = w^{n+1} - w^n$, the numerical fluxes at time level $(n+1)$ can be expressed as follows :

$$\begin{aligned} F^{n+1} &= (\mu_1 f^E)^{n+1} - d_1(w^{n+1}) \\ &= \mu_1 f^E(w^n + \Delta w^n) - d_1(w^n + \Delta w^n) \quad , \\ G^{n+1} &= (\mu_2 g^E)^{n+1} - d_2(w^{n+1}) \\ &= \mu_2 g^E(w^n + \Delta w^n) - d_2(w^n + \Delta w^n) \quad . \end{aligned} \tag{1.2.11}$$

Using now the Jacobian matrices A^E, B^E and linearizing the above expressions around time level n yields :

$$\begin{aligned} F^{n+1} &\approx \mu_1 (f^E)^n + (A^E)^n \mu_1 \Delta w^n - d_1(w^n) - D_1^n \delta_1 \Delta w^n \ , \\ G^{n+1} &\approx \mu_2 (g^E)^n + (B^E)^n \mu_2 \Delta w^n - d_2(w^n) - D_2^n \delta_2 \Delta w^n \ . \end{aligned} \tag{1.2.12}$$

where the dissipation matrices D_1 and D_2 are "frozen" at time level n. Eventually, the linearized numerical fluxes are given by :

$$\begin{aligned} F^{n+1} &\approx F^n + (A^E)^n \mu_1 \Delta w^n - D_1^n \delta_1 \Delta w^n \ , \\ G^{n+1} &\approx G^n + (B^E)^n \mu_2 \Delta w^n - D_2^n \delta_2 \Delta w^n \ . \end{aligned} \tag{1.2.13}$$

Inserting these expressions into the implicit stage (1.2.10) yields :

$$\mathcal{H} \cdot \Delta w_{i,j}^n \ = \ -\Delta t \cdot \mathcal{R}_{i,j}^n \ , \tag{1.2.14}$$

where the right hand side $\mathcal{R}_{i,j}^n$ and the operator $\mathcal{H}$ associated with the implicit stage are defined by :

$$\left\{ \begin{array}{l} \mathcal{R}_{i,j}^n = \left(\dfrac{\delta_1 F}{\delta x} + \dfrac{\delta_2 G}{\delta y} \right)_{i,j}^n \ , \\ \mathcal{H} = \left(Id + \sigma_1 \delta_1 (A^E)^n \mu_1 + \sigma_2 \delta_2 (B^E)^n \mu_2 - \sigma_1 \delta_1 D_1^n \delta_1 - \sigma_2 \delta_2 D_2^n \delta_2 \right) \ , \end{array} \right. \tag{1.2.15}$$

with Id the identity operator. The left hand side of equation (1.2.14) can also be written as follows :

$$\mathcal{H} \cdot \Delta w_{i,j}^n = C_0 \Delta w_{i,j}^n + C_1^- \Delta w_{i-1,j}^n + C_1^+ \Delta w_{i+1,j}^n + C_2^- \Delta w_{i,j-1}^n + C_2^+ \Delta w_{i,j+1}^n \ . \tag{1.2.16}$$

where the coefficients $C_p^{\pm}$, C_0 are given by :

$$\left\{ \begin{array}{l} C_1^- = -\sigma_1 \left(\frac{1}{2} A^E + D_1 \right)_{i-\frac{1}{2},j}^n = -\sigma_1 \left(\mathcal{A}^+ \right)_{i-\frac{1}{2},j}^n \\ C_1^+ = \sigma_1 \left(\frac{1}{2} A^E - D_1 \right)_{i+\frac{1}{2},j}^n \quad = \sigma_1 \left(\mathcal{A}^- \right)_{i+\frac{1}{2},j}^n \\ C_2^- = -\sigma_2 \left(\frac{1}{2} B^E + D_2 \right)_{i,j-\frac{1}{2}}^n = -\sigma_2 \left(\mathcal{B}^+ \right)_{i,j-\frac{1}{2}}^n \\ C_2^+ = \sigma_2 \left(\frac{1}{2} B^E - D_2 \right)_{i,j+\frac{1}{2}}^n \quad = \sigma_2 \left(\mathcal{B}^- \right)_{i,j+\frac{1}{2}}^n \\ C_0 = Id + \sigma_1 \left(\mathcal{A}^+ \right)_{i+\frac{1}{2},j}^n - \sigma_1 \left(\mathcal{A}^- \right)_{i-\frac{1}{2},j}^n + \sigma_2 \left(\mathcal{B}^+ \right)_{i,j+\frac{1}{2}}^n - \sigma_2 \left(\mathcal{B}^- \right)_{i,j-\frac{1}{2}}^n \end{array} \right. \tag{1.2.17}$$

Applying the implicit scheme (1.2.14) at each point of the computational domain (without taking into account the boundary conditions) yields the following linear system :

$$M \cdot [\Delta w^n] = [\mathcal{R}^n] \ , \tag{1.2.18}$$

with a left-hand-side (LHS) matrix defined by

$$M = \begin{bmatrix} x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot \\ x & \cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot & \\ \cdot & C_2^- & \cdot & \cdot & C_1^- & C^0 & C_1^+ & \cdot & \cdot & C_2^+ & \cdot \\ \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot & x \\ \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x \end{bmatrix} \ , \tag{1.2.19}$$

where x denotes a non-zero generic coefficient, all the dotted spots corresponding to null coefficients; $[\Delta w^n]$ and $[\mathcal{R}^n]$ denote respectively the vector of unknown implicit increments and know explicit increments :

$$[\Delta w^n] = \begin{bmatrix} . \\ \Delta w^n_{i,j-1} \\ . \\ . \\ \Delta w^n_{i-1,j} \\ \Delta w^n_{i,j} \\ \Delta w^n_{i+1,j} \\ . \\ . \\ \Delta w^n_{i,j+1} \\ . \end{bmatrix} \quad , \quad [\mathcal{R}^n] = -\Delta t \begin{bmatrix} . \\ \mathcal{R}^n_{i,j-1} \\ . \\ . \\ \mathcal{R}^n_{i-1,j} \\ \mathcal{R}^n_{i,j} \\ \mathcal{R}^n_{i+1,j} \\ . \\ . \\ \mathcal{R}^n_{i,j+1} \\ . \end{bmatrix} \quad . \tag{1.2.20}$$

The scheme defined by (1.2.18),(1.2.19), (1.2.20) is referred to as a *block implicit scheme* because all the coefficients of the matrix M are 4×4 (respectively 5×5) matrices for $2D$ (respectively $3D$) problems. Time-advancement using the block implicit scheme requires to solve the linear system (1.2.18) at each iteration in order to obtain w^{n+1} from Δw^n ($w^{n+1} = w^n + \Delta w^n$). Hence, the efficiency of the block implicit scheme strongly depends on the method used to solve the linear system. It is important to point out a direct solution of (1.2.18) is seldom undertaken because of its excessive cost both in terms of computational time and memory storage. Alternatively, system (1.2.18) is usually solved in a approximate way taking advantage of some factorization of the LHS matrix M or resorting to some iterative process. The design of a numerical treatment offering a fast solution (in terms of CPU time) while requiring low memory constraints is at the very heart of the present work. The next chapter will be devoted to the presentation and discussion of some standard implicit treatments such as LU-decomposition, Approximate Factorization, Line Relaxation, Krylov Methods ... The following sections of this chapter deal with the description of some extensions of the implicit upwind scheme first presented for the Euler equations : higher-order extension, viscous case, low-Mach number regime and unsteady flows. As will be shown, the basic formulation (1.2.18) remains formally unchanged : introducing viscous effects, physical time-derivative and preconditioning into the implicit stage will simply lead to more involved expression of the block coefficients (1.2.17).

Remark

Even though not every scheme can be put in the assumed form (1.2.6), it is nevertheless not difficult to adapt the previous design steps of a block implicit stage to other forms of explicit numerical fluxes. For instance, let us consider the numerical fluxes formally expressed as $F_{i+\frac{1}{2},j} = F(w^L_{i+\frac{1}{2},j}, w^R_{i+\frac{1}{2},j})$ with $w^L_{i+\frac{1}{2},j} = w_{i,j}$, $w^R_{i+\frac{1}{2},j} = w_{i+1,j}$ for a first-order scheme and similarly $G_{i,j+\frac{1}{2}} = G(w^L_{i,j+\frac{1}{2}}, w^R_{i,j+\frac{1}{2}})$ with $w^L_{i,j+\frac{1}{2}} = w_{i,j}$, $w^R_{i,j+\frac{1}{2}} = w_{i,j+1}$. To be a little more specific, for a scheme of Flux Vector Splitting type (such as Van Leer scheme for instance), the

numerical fluxes will read :

$$\begin{cases} F(w^L_{i+\frac{1}{2},j}, w^R_{i+\frac{1}{2},j}) = F^+(w^L_{i+\frac{1}{2},j}) + F^-(w^R_{i+\frac{1}{2},j}) \\ \\ G(w^L_{i,j+\frac{1}{2}}, w^R_{i,j+\frac{1}{2}}) = G^+(w^L_{i,j+\frac{1}{2}}) + G^-(w^R_{i,j+\frac{1}{2}}) \end{cases}$$

Note that Roe and Rusanov schemes are obtained by taking :

$$\begin{cases} F(w^L_{i+\frac{1}{2},j}, w^R_{i+\frac{1}{2},j}) = \frac{1}{2}(f^E(w^L_{i+\frac{1}{2},j}) + f^E(w^R_{i+\frac{1}{2},j})) - (D_1)_{i+\frac{1}{2},j}(w^R_{i+\frac{1}{2},j} - w^L_{i+\frac{1}{2},j}) \\ \\ G(w^L_{i,j+\frac{1}{2}}, w^R_{i,j+\frac{1}{2}}) = \frac{1}{2}(g^E(w^L_{i,j+\frac{1}{2}}) + g^E(w^R_{i,j+\frac{1}{2}})) - (D_2)_{i,j+\frac{1}{2}}(w^R_{i,j+\frac{1}{2}} - w^L_{i,j+\frac{1}{2}}) \end{cases}$$

with the dissipation coefficients respectively given by (1.2.8) and (1.2.9).
The derivation of an implicit stage associated with the explicit scheme based on such expressions of the numerical fluxes follows the same steps than the process previously described. Taking the numerical fluxes at the new time-level, F^{n+1} and G^{n+1}, and linearizing these terms around time level n leads to :

$$\begin{cases} F((w^L)^{n+1}_{i+\frac{1}{2},j}, (w^R)^{n+1}_{i+\frac{1}{2},j}) = F^n_{i+\frac{1}{2},j} + \left(J^L_1\right)^n_{i+\frac{1}{2},j} \cdot (\Delta w^L)^n_{i+\frac{1}{2},j} + \left(J^R_1\right)^n_{i+\frac{1}{2},j} \cdot (\Delta w^R)^n_{i+\frac{1}{2},j} \quad , \\ \\ G((w^L)^{n+1}_{i,j+\frac{1}{2}}, (w^R)^{n+1}_{i,j+\frac{1}{2}}) = G^n_{i,j+\frac{1}{2}} + \left(J^L_2\right)^n_{i,j+\frac{1}{2}} \cdot (\Delta w^L)^n_{i,j+\frac{1}{2}} + \left(J^R_2\right)^n_{i,j+\frac{1}{2}} \cdot (\Delta w^R)^n_{i,j+\frac{1}{2}} \quad , \end{cases} \tag{1.2.21}$$

where $J^L_1 = \partial F/\partial w^L$, $J^R_1 = \partial F/\partial w^R$ and $J^L_2 = \partial G/\partial w^L$, $J^R_2 = \partial G/\partial w^R$ are the Jacobian matrices of the numerical fluxes F and G. The implicit scheme takes eventually the generic form (identical to (1.2.14)) :

$$\mathcal{H} \cdot \Delta w^n_{i,j} \;=\; -\Delta t \cdot \mathcal{R}^n_{i,j} \;,$$

with

$$\begin{cases} \mathcal{R}^n_{i,j} = \left(\dfrac{\delta_1 F}{\delta x} + \dfrac{\delta_2 G}{\delta y}\right)^n_{i,j} \quad , \\ \\ \mathcal{H} \cdot \Delta w^n_{i,j} = C_0 \Delta w^n_{i,j} + C^-_1 \Delta w^n_{i-1,j} + C^+_1 \Delta w^n_{i+1,j} + C^-_2 \Delta w^n_{i,j-1} + C^+_2 \Delta w^n_{i,j+1} \;, \end{cases}$$

where the matrix coefficients $C^\pm_p$, C_0 are given by :

$$\begin{cases} C^-_1 = -\sigma_1 (J^L_1)^n_{i-\frac{1}{2},j} \\ \\ C^+_1 = \sigma_1 (J^R_1)^n_{i+\frac{1}{2},j} \\ \\ C^-_2 = -\sigma_2 (J^L_2)^n_{i,j-\frac{1}{2}} \\ \\ C^+_2 = \sigma_2 (J^R_2)^n_{i,j+\frac{1}{2}} \\ \\ C_0 = Id + \sigma_1 (J^L_1)^n_{i+\frac{1}{2},j} - \sigma_1 (J^R_1)^n_{i-\frac{1}{2},j} + \sigma_2 (J^L_2)^n_{i,j+\frac{1}{2}} - \sigma_2 (J^R_2)^n_{i,j-\frac{1}{2}} \end{cases}$$

It is easy to check the above expressions give back (1.2.17) when the explicit stage is based on the Roe or Rusanov scheme. For other numerical flux formulas, alternative expressions will be found for the coefficients $C^\pm_p$ but, clearly, whatever the numerical flux retained when writing the explicit stage, the resulting implicit scheme will require the solution of a block linear system.

1.2.2 High-order extension

The upwind scheme presented in the previous section is based on a compact stencil making use of three points in each space direction, which leads to a first-order accurate upwind scheme. This accuracy order is not sufficient for practical purposes (where the grid resolution is not necessarily very high) so that it is customary to increase the space accuracy of an upwind scheme (be it of Flux Vector Splitting, Flux Difference Splitting or Hybrid type - AUSM+, CUSP schemes) by resorting to the well-known variable reconstruction MUSCL (Monotone Upwind Scheme for Conservative Law) approach originally introduced by Van Leer [78]. Using this technique, the numerical flux formulas previously introduced and based on a left w^L and right w^R states remain unchanged but these left and right states are now approximated with an increased accuracy thanks to an extrapolation using the values of w available on an enlarged stencil (*cf.* figure 1.2.3) :

$$\begin{aligned}
w^L_{i+1/2,j} &= w^n_{i,j} + \tfrac{\varepsilon}{4}(1-\kappa)(w^n_{i,j} - w^n_{i-1,j}) + \tfrac{\varepsilon}{4}(1+\kappa)(w^n_{i+1,j} - w^n_{i,j}) \\
w^R_{i+1/2,j} &= w^n_{i+1,j} - \tfrac{\varepsilon}{4}(1+\kappa)(w^n_{i+1,j} - w^n_{i,j}) - \tfrac{\varepsilon}{4}(1-\kappa)(w^n_{i+2,j} - w^n_{i+1,j}) \\
w^L_{i,j-1/2} &= w^n_{i,j-1} + \tfrac{\varepsilon}{4}(1-\kappa)(w^n_{i,j-1} - w^n_{i,j-2}) + \tfrac{\varepsilon}{4}(1+\kappa)(w^n_{i,j} - w^n_{i,j-1}) \\
w^R_{i,j-1/2} &= w^n_{i,j} - \tfrac{\varepsilon}{4}(1+\kappa)(w^n_{i,j} - w^n_{i,j-1}) - \tfrac{\varepsilon}{4}(1-\kappa)(w^n_{i,j+1} - w^n_{i,j})
\end{aligned} \tag{1.2.22}$$

where the choice $\varepsilon = 0$ gives back the first-order schemes while $\varepsilon = 1$ activates the reconstruction process. The parameter κ allows to control the order of accuracy of the reconstruction as will be soon explained. In the case of the Roe or Rusanov schemes for instance, the numerical fluxes

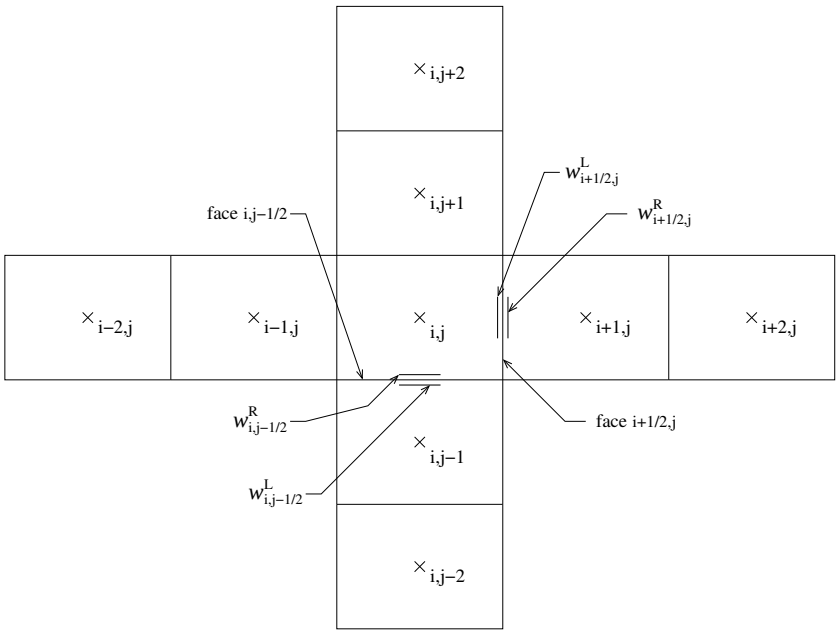


Figure 1.2.3: Enlarged stencil for 2nd-order space accurate upwind scheme

are then evaluated using the previously introduced formulas :

$$\begin{aligned} F(w^L_{i+\frac{1}{2},j}, w^R_{i+\frac{1}{2},j}) &= \frac{1}{2}\left(f^E(w^L_{i+\frac{1}{2},j}) + f^E(w^R_{i+\frac{1}{2},j})\right) - D_1(\bar{w}_{i+\frac{1}{2},j})\left(w^R_{i+\frac{1}{2},j} - w^L_{i+\frac{1}{2},j}\right) \\ G(w^L_{i,j-\frac{1}{2}}, w^R_{i,j-\frac{1}{2}}) &= \frac{1}{2}\left(g^E(w^L_{i,j-\frac{1}{2}}) + g^E(w^R_{i,j-\frac{1}{2}})\right) - D_2(\bar{w}_{i,j-\frac{1}{2}})\left(w^R_{i,j-\frac{1}{2}} - w^L_{i,j-\frac{1}{2}}\right) \end{aligned} \tag{1.2.23}$$

where $\bar{w}$ denotes an average state (Roe average for instance) computed at the face using the left and right states w^L, w^R. In the linear case, where $f^E = A^E w$, $g^E = B^E w$ with A^E, B^E constant matrices, it is easy to show formulas (1.2.23) can be developed as :

$$\begin{aligned} F^n_{i+\frac{1}{2},j} &= \left((\mu_1 f^E) - \frac{\varepsilon}{4}(1-\kappa)(\delta_1^2\mu_1 f^E)\right)^n_{i+\frac{1}{2},j} - \underbrace{\left((1-\varepsilon)D_1\delta_1 w - \frac{\varepsilon}{4}(1-\kappa)D_1\delta_1^3 w\right)^n_{i+\frac{1}{2},j}}_{d_1} \\ G^n_{i-\frac{1}{2},j} &= \left((\mu_2 g^E) - \frac{\varepsilon}{4}(1-\kappa)(\delta_2^2\mu_2 g^E)\right)^n_{i-\frac{1}{2},j} - \underbrace{\left((1-\varepsilon)D_2\delta_2 w - \frac{\varepsilon}{4}(1-\kappa)D_2\delta_2^3 w\right)^n_{i-\frac{1}{2},j}}_{d_2} \end{aligned} \tag{1.2.24}$$

and eventually yield the following explicit residual (approximating the flux balance $f^E_x + g^E_y$) :

$$\begin{aligned} \mathcal{R}^n_{i,j} = &\frac{\delta_1\mu_1(f^E)^n_{i,j}}{\delta x} - \frac{\varepsilon}{4}(1-\kappa)\frac{\delta_1^3\mu_1(f^E)^n_{i,j}}{\delta x} \\ &- (1-\varepsilon)\delta x\frac{\delta_1(D_1\delta_1 w)^n_{i,j}}{\delta x^2} - \frac{\varepsilon}{4}(1-\kappa)\delta x^3\frac{\delta_1(D_1\delta_1^3 w)^n_{i,j}}{\delta x^4} \\ &+ \frac{\delta_2\mu_2(g^E)^n_{i,j}}{\delta y} - \frac{\varepsilon}{4}(1-\kappa)\frac{\delta_2^3\mu_2(g^E)^n_{i,j}}{\delta y} \\ &- (1-\varepsilon)\delta y\frac{\delta_2(D_2\delta_2 w)^n_{i,j}}{\delta y^2} - \frac{\varepsilon}{4}(1-\kappa)\delta y^3\frac{\delta_2(D_2\delta_2^3 w)^n_{i,j}}{\delta y^4} \ . \end{aligned} \tag{1.2.25}$$

As already pointed out, the choice $\varepsilon = 0$ allows to recover the initial first-order formulation (1.2.6). If $\varepsilon = 1$, the numerical dissipative fluxes d_1, d_2 become 3rd-order terms so that the resulting dissipation in the above expression is based on fourth-derivatives multiplied by some positive coefficient (in order to ensure the scheme is truly dissipative); meanwhile, the non-dissipative fluxes remain in general approximated at second order using simply centered formulas. However, in the particular case $\kappa = 1/3$, the reconstruction process allows to cancel the dispersive error of the central average leading to a 4th-order centered approximation of the non-dissipative fluxes, completed of course with the third-order dissipation, thus yielding a globally 3rd-order accurate upwind scheme :

$$\begin{aligned} \mathcal{R}^n_{i,j} = &\frac{\delta_1\mu_1(Id - \frac{1}{6}\delta_1^2)(f^E)^n_{i,j}}{\delta x} + \frac{1}{6}\delta x^3\frac{\delta_1(D_1\delta_1^3 w)^n_{i,j}}{\delta x^4} \\ &+ \frac{\delta_2\mu_2(Id - \frac{1}{6}\delta_2^2)(g^E)^n_{i,j}}{\delta y} + \frac{1}{6}\delta y^3\frac{\delta_2(D_2\delta_2^3 w)^n_{i,j}}{\delta y^4} \ . \end{aligned} \tag{1.2.26}$$

There is basically two ways to make the above Roe-MUSCL or Rusanov-MUSCL scheme implicit, that both lead anyway to an implicit scheme of the form :

$$\mathcal{H} \cdot \Delta w_{i,j}^n \;=\; -\Delta t \cdot (\mathcal{R}^{III})_{i,j}^n \;, \tag{1.2.27}$$

where the notation $\mathcal{R}^{III}$ emphasizes the fact a third-order reconstruction ($\varepsilon = 1$, $\kappa = 1/3$) has been used in the explicit stage. For the sake of simplicity in the presentation, let us assume scheme (1.2.27) is applied to a linear problem with $f^E = A^E w$, $g^E = B^E w$, A^E and B^E being constant matrices. Under this assumption, the residual $\mathcal{R}^{III}$ reads (when the Roe flux formula is used) :

$$(\mathcal{R}^{III})_{i,j}^n = A^E \frac{\delta_1 \mu_1 (Id - \frac{1}{6}\delta_1^2) w_{i,j}^n}{\delta x} + \frac{1}{12}|A^E| \frac{\delta_1^4 w_{i,j}^n}{\delta x}$$

$$+ \; B^E \frac{\delta_2 \mu_2 (Id - \frac{1}{6}\delta_2^2) w_{i,j}^n}{\delta y} + \frac{1}{12}|B^E| \frac{\delta_2^4 w_{i,j}^n}{\delta y}$$

or, since the problem is linear, $-\Delta t (\mathcal{R}^{III})_{i,j}^n = -\mathcal{K}^{III} w_{i,j}^n$ with a third-order explicit stage discretization operator given by :

$$\mathcal{K}^{III} = \dot{A}^E \delta_1 \mu_1 (Id - \frac{1}{6}\delta_1^2) + \dot{B}^E \delta_2 \mu_2 (Id - \frac{1}{6}\delta_2^2) + \frac{1}{12}|\dot{A}^E| \delta_1^4 + \frac{1}{12}|\dot{B}^E| \delta_2^4,$$

where $\dot{A}^E = \frac{\Delta t}{\delta x} A^E$, $\dot{B}^E = \frac{\Delta t}{\delta y} B^E$. Using a first-order discretization yields $-\Delta t (\mathcal{R}^I)_{i,j}^n = -\mathcal{K}^I w_{i,j}^n$ with :

$$\mathcal{K}^I = \dot{A}^E \delta_1 \mu_1 + \dot{B}^E \delta_2 \mu_2 - \frac{1}{2}|\dot{A}^E| \delta_1^2 - \frac{1}{2}|\dot{B}^E| \delta_2^2,$$

Starting from a first-order explicit stage, it is natural to derive a first-order upwind implicit scheme of the form :

$$\mathcal{H}^I \cdot \Delta w_{i,j}^n \;=\; -\mathcal{K}^I w_{i,j}^n \tag{1.2.28}$$

with $\mathcal{H}^I = Id + \mathcal{K}^I$ and this expression corresponds precisely to the linear version of the previously derived implicit stage (1.2.15); such a scheme will be referred to as I/I to recall a first-order implicit stage is associated to a first-order explicit stage.
Starting now from a third-order explicit stage, a direct approach would lead to a third-order upwind implicit scheme of the form :

$$\mathcal{H}^{III} \cdot \Delta w_{i,j}^n \;=\; -\mathcal{K}^{III} w_{i,j}^n \tag{1.2.29}$$

with $\mathcal{H}^{III} = Id + \mathcal{K}^{III}$. However, it is easy to check the linear system associated with such an implicit stage is much more involved than the one obtained with a simple first-order implicit stage since it couples now 5 (instead of 3) unknown time-increments in each grid-direction (in other words the bandwidth of the block matrix M defined in (1.2.19) is increased). Even though the choice (1.2.29), denoted III/III from now on to recall a third-order implicit stage is associated with a third-order explicit stage, is judicious in terms of intrinsic efficiency - *i.e.* it provides a strong damping of all the error modes at each time iteration (this point will be demonstrated using Fourier analysis in chapter 3) -, the necessity to solve a more complex linear system is likely

to increase the cost per iteration of this implicit scheme in such a proportion that alternative simpler choices eventually become more interesting. Precisely, a strategy followed by many authors [64, 12], and also retained in this work, is to associate a simple first-order implicit stage to a third-order explicit stage :

$$\mathcal{H}^I \cdot \Delta w_{i,j}^n \;=\; -\mathcal{K}^{III} w_{i,j}^n \tag{1.2.30}$$

where the implicit and explicit discretization operators used in this I/III strategy have been previously defined. As will be seen using Fourier analysis, such a choice leads naturally to a lower intrinsic efficiency but this loss of error damping is more than balanced by the reduced cost taken by the solution of a simple first-order implicit stage instead of a more involved third-order implicit stage.

Remark

In the case of the Euler equations, the inviscid fluxes f^E and g^E are non-linear functions of w, so that the choice $\kappa = 1/3$ cannot ensure a true 3rd-order accuracy (a reconstruction process applied on the fluxes would circumvent this problem). Nevertheless, since this simple choice of variable reconstruction allows anyway to increase the effective accuracy of an upwind scheme, it is commonly retained and will be systematically used for the calculations presented in this work. Note also several choices are possible for the vector of variables on which to apply the reconstruction process. In practice, the set of primitive variables $(\rho,\ u,\ v,\ p)^T$ is typically used rather than the set of conservative variables $(\rho,\ \rho u,\ \rho v,\ \rho E)^T$.

1.2.3 Extension to Navier-Stokes equations

Consider now the two-dimensional Navier-Stokes equations :

$$w_t \;+\; (f^E(w) - f^V(w, w_x, w_y))_x \;+\; (g^E(w) - g^V(w, w_x, w_y))_y \;=\; 0 \tag{1.2.31}$$

where the notations used for f^V and g^V emphasize the viscous fluxes depend not only on the conservative variables w but also on their gradients. All the schemes considered in the present work rely on a separate treatment for the inviscid and viscous fluxes. Namely, the discretization of the inviscid fluxes is carried out in the way described in the previous section. The numerical approximation of the viscous fluxes is purely centered and second-order accurate; it is obtained from the following steps. When discretizing (1.2.31) at point (i, j) of a regular Cartesian grid with space steps δx and δy (of the same order h), the numerical approximation of f^V and g^V requires to evaluate the vector w and the space derivatives w_x and w_y on each face of the control cell (see Fig. 1.2.4). In other words, the viscous numerical fluxes F^V and G^V can be expressed as :

$$\begin{aligned} F^V_{i+\frac{1}{2},j} &= f^V(w_{i+\frac{1}{2},j}, (w_x)_{i+\frac{1}{2},j}, (w_y)_{i+\frac{1}{2},j}) \\ G^V_{i,j-\frac{1}{2}} &= g^V(w_{i,j-\frac{1}{2}}, (w_x)_{i,j-\frac{1}{2}}, (w_y)_{i,j-\frac{1}{2}}) \end{aligned}$$

where it remains to evaluate the quantities $w_{i+\frac{1}{2},j}$, $(w_x)_{i+\frac{1}{2},j}$, $(w_y)_{i+\frac{1}{2},j}$ and $w_{i,j-\frac{1}{2}}$, $(w_x)_{i,j-\frac{1}{2}}$, $(w_y)_{i,j-\frac{1}{2}}$). This evaluation is performed using simple second-order centered approximation (since the isotropy of the diffusion phenomena does not call for any kind of upwinding). Let us consider

for instance the face $(i+1/2,j)$ where the evaluation is carried out as follows :

$$\begin{aligned}(w)_{i+1/2,j} &= \frac{w_{i,j}+w_{i+1,j}}{2}+O(h^2)\\ (w_x)_{i+1/2,j} = \left.\frac{\partial w}{\partial x}\right|_{i+1/2,j} &= \frac{w_{i,j}-w_{i+1,j}}{\delta x}+O(h^2)\\ (w_y)_{i+1/2,j} = \left.\frac{\partial w}{\partial y}\right|_{i+1/2,j} &= \frac{w_{i+1/2,j+1/2}-w_{i+1/2,j-1/2}}{\delta y}+O(h^2)\end{aligned} \tag{1.2.32}$$

In the last formula, the states needed at nodes $(i+1/2,j+1/2)$ and $(i+1/2,j-1/2)$ to compute

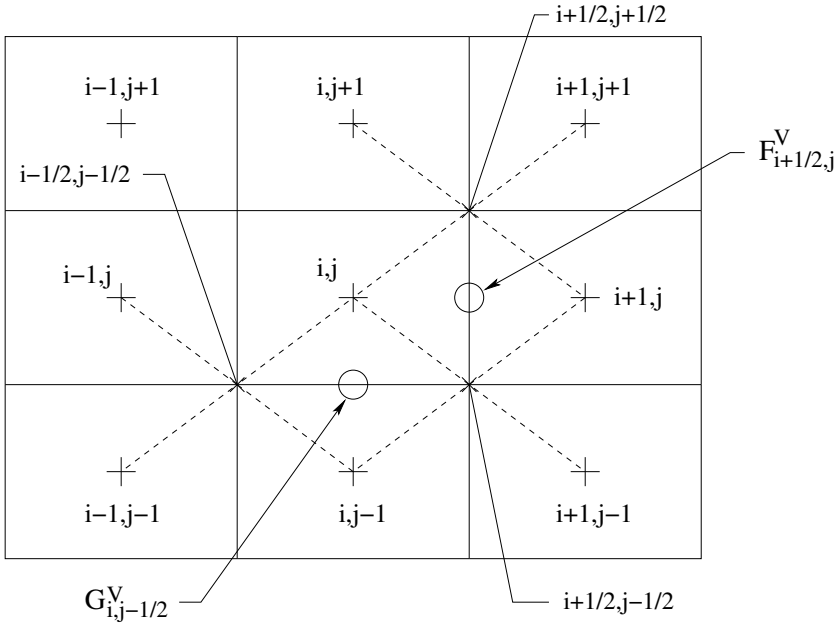


Figure 1.2.4: Stencil for the discretization of the viscous fluxes.

the space derivative w_y on the face $(i+\frac{1}{2},j)$ are obtained by taking the average of the states in the four neighboring cells (*cf.* figure 1.2.4) :

$$(w)_{i+1/2,j+1/2} = \frac{w_{i,j}+w_{i+1,j}+w_{i+1,j+1}+w_{i,j+1}}{4}+O(h^2) = (\mu_1\mu_2 w)_{i+1/2,j+1/2}+O(h^2)$$

Thus the vector w and the space derivatives w_x and w_y evaluated on the face $(i+1/2,j)$ read finally :

$$\begin{aligned}(w)_{i+1/2,j} &= (\mu_1 w)_{i+1/2,j}+O(h^2)\\ (w_x)_{i+1/2,j} &= (\frac{\delta_1 w}{\delta x})_{i+1/2,j}+O(h^2)\\ (w_y)_{i+1/2,j} &= (\frac{\delta_2\mu_1\mu_2 w}{\delta y})_{i+1/2,j}+O(h^2)\end{aligned} \tag{1.2.33}$$

The expression of the numerical viscous fluxes F^V and G^V can be summarized as :

$$\begin{aligned} F^V_{i+1/2,j} &= f^V((\mu_1 w)_{i+1/2,j}, (\frac{\delta_1 w}{\delta x})_{i+1/2,j}, (\frac{\delta_2\mu_2\mu_1 w}{\delta y})_{i+1/2,j}) \\ G^V_{i,j-1/2} &= g^V((\mu_2 w)_{i,j-1/2}, (\frac{\delta_1\mu_1\mu_2 w}{\delta x})_{i,j-1/2}, (\frac{\delta_2 w}{\delta y})_{i,j-1/2}) \end{aligned} \tag{1.2.34}$$

Finally, the full numerical fluxes (including the inviscid and viscous contributions) can be written as follows :

$$\begin{aligned} F_{i+1/2,j} &= F^E(w^L_{i+1/2,j}, w^R_{i+1/2,j}) - f^V((\mu_1 w)_{i+1/2,j}, \frac{(\delta_1 w)_{i+1/2,j}}{\delta x}, \frac{(\delta_2\mu_1\mu_2 w)_{i+1/2,j}}{\delta y}) \\ G_{i,j-1/2} &= G^E(w^L_{i,j-1/2}, w^R_{i,j-1/2}) - g^V((\mu_2 w)_{i,j-1/2}, \frac{(\delta_1\mu_2\mu_1 w)_{i,j-1/2}}{\delta x}, \frac{(\delta_2 w)_{i,j-1/2}}{\delta y}) \end{aligned} \tag{1.2.35}$$

where the exact expression of F^E and G^E depends on the choice of the inviscid numerical flux formula (Roe, Rusanov, AUSM+, ...)

Implicit treatment

Deriving an implicit version of the scheme defined by the explicit stage :

$$\Delta w^{exp}_{i,j} = -\Delta t(\frac{\delta_1 F}{\delta x} + \frac{\delta_2 G}{\delta y})_{i,j}$$

with F and G given by (1.2.35) supposes to build an approximation of the viscous fluxes F^V, G^V at the new time level $(n+1)$. Performing a Taylor development of these fluxes around $(n+1)\Delta t$ and making use of the Jacobian matrices A^V_0, A^V_1, A^V_2 and B^V_0, B^V_1, B^V_2 introduced in the first section of this chapter (see (1.1.14)) leads to :

$$\begin{aligned} (F^V)^{n+1} &= (F^V)^n + \Delta t\ A^V_0\ \frac{\partial w}{\partial t} + \Delta t\ A^V_1\ \frac{\partial w_x}{\partial t} + \Delta t\ A^V_2\ \frac{\partial w_y}{\partial t} + O(\Delta t^2) \\ (G^V)^{n+1} &= (G^V)^n + \Delta t\ B^V_0\ \frac{\partial w}{\partial t} + \Delta t\ B^V_1\ \frac{\partial w_x}{\partial t} + \Delta t\ B^V_2\ \frac{\partial w_y}{\partial t} + O(\Delta t^2) \end{aligned} \tag{1.2.36}$$

so that a centered approximation of the terms appearing in these developments yields the following discrete formulas :

$$\begin{aligned} (F^V)^{n+1}_{i+\frac{1}{2},j} = (F^V)^n_{i+\frac{1}{2},j} &+ (A^V_0)_{i+\frac{1}{2},j}\ \mu_1\Delta w^n_{i+\frac{1}{2},j} \\ &+ (A^V_1)_{i+\frac{1}{2},j}\ \frac{\delta_1\Delta w^n_{i+\frac{1}{2},j}}{\delta x} \\ &+ (A^V_2)_{i+\frac{1}{2},j}\ \frac{\delta_2\mu_2\mu_1\Delta w^n_{i+\frac{1}{2},j}}{\delta y} + O(\Delta t^2, h^2) \\ (G^V)^{n+1}_{i,j-\frac{1}{2}} = (G^V)^n_{i,j-\frac{1}{2}} &+ (B^V_0)_{i,j-\frac{1}{2}}\ \mu_2\Delta w^n_{i,j-\frac{1}{2}} \\ &+ (B^V_1)_{i,j-\frac{1}{2}}\ \frac{\delta_1\mu_1\mu_2\Delta w^n_{i,j-\frac{1}{2}}}{\delta x} \\ &+ (B^V_2)_{i,j-\frac{1}{2}}\ \frac{\delta_2\Delta w^n_{i,j-\frac{1}{2}}}{\delta y} + O(\Delta t^2, h^2) \end{aligned} \tag{1.2.37}$$

In order to obtain a viscous contribution to the implicit stage that remains as simple as the first-order upwind contribution coming from the inviscid numerical fluxes, the following simplified development is retained :

$$\begin{aligned}(F^V)^{n+1}_{i+1/2,j} &\approx (F^V)^n_{i+1/2,j} + (A_1^V)^n_{i+1/2,j}\frac{(\delta_1\Delta w)^n_{i+1/2,j}}{\delta x}\\ (G^V)^{n+1}_{i,j-1/2} &\approx (G^V)^n_{i,j-1/2} + (B_2^V)^n_{i,j-1/2}\frac{(\delta_2\Delta w)^n_{i,j-1/2}}{\delta y}\end{aligned} \tag{1.2.38}$$

The terms containing the viscous Jacobian matrices A_2^V and B_1^V are discarded because they enlarge the stencil of the implicit stage with respect to a basic 3-point per direction support; as for the terms containing the viscous Jacobian matrices A_0^V, B_0^V they are not retained because the evaluation of these matrices is quite expensive, since it requires to compute components of the gradient of w. Finally, the block-implicit upwind scheme used to approximate the Navier-Stokes equations remains formally unchanged with respect to the case of the Euler equations :

$$\mathcal{H}\cdot\Delta w^n_{i,j} \;=\; -\Delta t\cdot\mathcal{R}^n_{i,j}\;,$$

with

$$\left\{\begin{array}{l}\mathcal{R}^n_{i,j} = \left(\dfrac{\delta_1(F^E-F^V)}{\delta x}+\dfrac{\delta_2(G^E-G^V)}{\delta y}\right)^n_{i,j}\;,\\ \mathcal{H} = \left(Id+\sigma_1\delta_1(A^E)^n\mu_1+\sigma_2\delta_2(B^E)^n\mu_2-\sigma_1\delta_1(D_1+\dfrac{A_1^V}{\delta x})^n\delta_1-\sigma_2\delta_2(D_2+\dfrac{B_2^V}{\delta y})^n\delta_2\right)\;.\end{array}\right. \tag{1.2.39}$$

The LHS of equation (1.2.39) can also be expanded as follows :

$$\mathcal{H}\cdot\Delta w^n_{i,j} = C_0\Delta w^n_{i,j} + C_1^-\Delta w^n_{i-1,j} + C_1^+\Delta w^n_{i+1,j} + C_2^-\Delta w^n_{i,j-1} + C_2^+\Delta w^n_{i,j+1}\;, \tag{1.2.40}$$

where the matrix coefficients $C_p^{\pm}$, C_0 now include a viscous contribution :

$$\left\{\begin{array}{ll}C_1^- = -\sigma_1\left(\frac{1}{2}A^E+D_1+\frac{1}{\delta x}A_1^V\right)^n_{i-\frac{1}{2},j} & = -\sigma_1\left(\mathcal{A}^+\right)^n_{i-\frac{1}{2},j}\\ C_1^+ = \sigma_1\left(\frac{1}{2}A^E-D_1-\frac{1}{\delta x}A_1^V\right)^n_{i+\frac{1}{2},j} & = \sigma_1\left(\mathcal{A}^-\right)^n_{i+\frac{1}{2},j}\\ C_2^- = -\sigma_2\left(\frac{1}{2}B^E+D_2+\frac{1}{\delta y}B_2^V\right)^n_{i,j-\frac{1}{2}} & = -\sigma_2\left(\mathcal{B}^+\right)^n_{i,j-\frac{1}{2}}\\ C_2^+ = \sigma_2\left(\frac{1}{2}B^E-D_2-\frac{1}{\delta y}B_2^V\right)^n_{i,j+\frac{1}{2}} & = \sigma_2\left(\mathcal{B}^-\right)^n_{i,j+\frac{1}{2}}\\ C_0 = Id+\sigma_1\left(\mathcal{A}^+\right)^n_{i+\frac{1}{2},j}-\sigma_1\left(\mathcal{A}^-\right)^n_{i-\frac{1}{2},j}+\sigma_2\left(\mathcal{B}^+\right)^n_{i,j+\frac{1}{2}}-\sigma_2\left(\mathcal{B}^-\right)^n_{i,j-\frac{1}{2}} & \end{array}\right. \tag{1.2.41}$$

Applying the viscous implicit scheme (1.2.39) at each point of the computational domain, one recovers the linear block system (1.2.18) where the coefficients of the matrix M and the residual $[\mathcal{R}^n]$ take into account the viscous terms as detailed in the above formulas.

1.3 Low-Mach Number Flows

1.3.1 Context

In view of the particular applications the CEA is interested in, namely the simulation of buoyant multicomponent reactive flow in nuclear reactor containment, the schemes developed in the present work should be versatile enough to deal with flow regimes ranging from nearly incompressible to highly compressible. It has been explained in the introduction of this thesis that two distinct strategies can be adopted in that case : either the extension of incompressible (pressure-based) flow solvers to non-zero Mach number flows or the extension of compressible (density-based) flows solvers to low-Mach number. It has been known for more than a decade now that a proper preconditioning, so-called low-Mach preconditioning, of the equations governing compressible flows enables the application of the schemes initially developed for compressible flows to the simulation of nearly incompressible flow problems and this choice is retained in our work for reasons detailed in the introduction (mainly because of the importance of existing developments on compressible solvers in CAST3M).
The implementation of a preconditioning method requires the preliminary choice of a set of variables in terms of which the Euler equations will be expressed before introducing a preconditioning. In fact, some of the preconditioning strategies proposed in the literature are identical but built using different sets of variables. For instance, Turkel [75] developed the first-version of his preconditioning starting from the Euler equations expressed in terms of entropic variables $(p,\ u,\ v,\ S)^T$ whereas the preconditioning strategy proposed by Weiss and Smith [85] is developed from the Euler and Navier-Stokes equations written for the primitive variables $\underline{q} = (p,\ u,\ v,\ T)^T$, also called viscous variables since viscous terms can be easily expressed using them, but turns out to be exactly the one proposed by Turkel when expressed with the same set of entropic variables. Note that Van Leer [80] developed his own preconditioning technique also using entropic variables $(p,\ u,\ v,\ S)^T$ whereas Choi and Merkle [14] or Venkateswaran *et al.* [81] employed primitive or viscous variables $(p,\ u,\ v,\ T)^T$. Note however, it is always the conservative form of the Euler or Navier-Stokes equations that is solved in order to ensure correct shock-capturing properties when the space-discretization is applied. The choice of viscous variables to express a preconditioned block-implicit upwind scheme can be convenient for general fluids with equations of state written in terms of pressure and temperature and can also be interesting to reduce the number of algebraic operations during the computation. Nevertheless, it must be clear that these different choices do not affect the convergence performance of the preconditioned scheme. During the course of this thesis, the use of either the vector of conservative variables $\underline{w} = (\rho, \rho u, \rho v, \rho E)^T$ and its increment $\Delta\underline{w}$ or the primitive variables through $\underline{q}$ and $\Delta\underline{q}$ has been thoroughly investigated (see the report [40]) in order to determine whether it was worth switching from the current $\underline{w}$-based formulation in CAST3M to a $\underline{q}$-based formulation. The results obtained, not reported here for the sake of conciseness, proved that the unit cost reduction offered by the choice of the primitive variables was too modest to undertake such heavy developments (such findings are not necessarily in agreement with the study [30] for instance but this question is clearly very much implementation / code-dependent). Consequently, the description of the preconditioned upwind schemes will focus in the present work on a formulation based on the set of conserved variables w.

1.3.2 Low-Mach preconditioning for conservative system

The low-Mach preconditioning method used in this work is based on the Turkel preconditioner [75]. This preconditioner has been originally applied for the Euler equations expressed in terms of entropic variables $V = (p,\ u,\ v,\ S)^T$ (with $S = \ln(p/\rho^\gamma)$). In quasi-linear form, this system takes the following form :

$$P_e^{-1} \cdot V_t + A_e \cdot V_x + B_e \cdot V_y = 0 \tag{1.3.1}$$

where

$$A_e = \frac{df^E}{dV} = \begin{pmatrix} u & \rho c^2 & 0 & 0 \\ 1/\rho & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u \end{pmatrix} \quad , \qquad B_e = \frac{dg^E}{dV} = \begin{pmatrix} v & 0 & \rho c^2 & 0 \\ 0 & v & 0 & 0 \\ 1/\rho & 0 & v & 0 \\ 0 & 0 & 0 & v \end{pmatrix}$$

with c the speed of sound. As for the preconditioning matrix P_e, it is designed so as to scale the pressure equation only :

$$P_e = \begin{pmatrix} \beta^2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The parameter β^2 is chosen such that the eigenvalues of the preconditioned system keep the same order of magnitude when the Mach number becomes small. The original definition proposed by Turkel [75] is :

$$\beta^2 = \min(\max(M^2, M_\infty^2), 1) \tag{1.3.2}$$

where M_∞ denotes the reference Mach number (inflow Mach number for external flows or maximum Mach number within the field for internal flows). This definition ensures the robustness of the preconditioned scheme and turns the scheme into its non-preconditioned version when $M > 1$. This choice is often referred to as a "global cut-off" because it does not allow β to be less than the inflow Mach number. Some local definitions are available in the literature, such as the one given in [23], which introduces a dependence on local gradients of pressure :

$$\beta^2 = \min(\max(M^2, M_p^2), 1)$$

with

$$M_p^2 = \frac{|\delta p|}{\rho\ c^2}$$

Unfortunately, the convergence rate of the preconditioned scheme with such a definition can become quite poor. On the other hand, the accuracy increases with the local cut-off since β can be less than the inflow Mach number which allows to scale more properly the numerical dissipation (see next sections). However, both local and global definitions fail to ensure a good efficiency for viscous or unsteady flows. It is thus necessary to redefine the cut-off in these cases so as to preserve efficiency. In [81], Venkateswaran *et al* have proposed the following definition :

$$\beta^2 = \min(\max(M^2, \beta_v^2, \beta_u^2), 1)$$

where β_v and β_u denote some viscous and unsteady cut-off respectively. In chapters 6 and 7, the definition of these new parameters will be detailed and their efficiency will be studied.

In order to build a low-Mach preconditioning for the Euler equations written in terms of conservative variables :

$$w_t + f(w)_x + g(w)_y = 0 \ ,$$

the transformation matrices allowing to switch from entropic variables V to conservative variables $w = (\rho,\ \rho u,\ \rho v,\ \rho E)^T$ are needed :

$$\frac{\partial w}{\partial V} = \begin{pmatrix} 1/c^2 & 0 & 0 & -\rho/\gamma \\ u/c^2 & \rho & 0 & -\rho u/\gamma \\ v/c^2 & 0 & \rho & -\rho v/\gamma \\ H/c^2 & \rho u & \rho v & -\rho q^2/2\gamma \end{pmatrix} \quad ,$$

$$\frac{\partial V}{\partial w} = \begin{pmatrix} (\gamma-1)q^2/2 & -u(\gamma-1) & -v(\gamma-1) & \gamma-1 \\ -u/\rho & 1/\rho & 0 & 0 \\ -v/\rho & 0 & 1/\rho & 0 \\ \frac{(\gamma-1)q^2}{2p} - \frac{\gamma}{\rho} & -\frac{u(\gamma-1)}{p} & -\frac{v(\gamma-1)}{p} & \frac{(\gamma-1)}{p} \end{pmatrix}$$

where $q^2 = u^2 + v^2$. Introducing these transformations into equation (1.3.1) yields :

$$P_e^{-1} \cdot \frac{\partial V}{\partial w} \cdot w_t + A_e \cdot \frac{\partial V}{\partial w} \cdot w_x + B_e \cdot \frac{\partial V}{\partial w} \cdot w_y = 0 \tag{1.3.3}$$

and left-multiplying by $\dfrac{\partial w}{\partial V}$ leads to :

$$\frac{\partial w}{\partial V} \cdot P_e \cdot \frac{\partial V}{\partial w} \cdot w_t + \frac{\partial w}{\partial V} \cdot A_e \cdot \frac{\partial V}{\partial w} \cdot w_x + \frac{\partial w}{\partial V} \cdot B_e \cdot \frac{\partial V}{\partial w} \cdot w_y = 0 \tag{1.3.4}$$

Therefore, the preconditioning matrix for the Euler equations in conservative variables can be defined as :

$$P_w = \frac{\partial w}{\partial v} \cdot P_e \cdot \frac{\partial v}{\partial w} \tag{1.3.5}$$

while the usual (conservative) Jacobian matrices of the convective fluxes are recovered by :

$$A^E = \frac{df^E}{dw} = A_w = \frac{\partial w}{\partial V} \cdot A_e \cdot \frac{\partial V}{\partial w} = \frac{\partial f}{\partial w} \quad , \quad B^E = \frac{dg^E}{dw} = B_w = \frac{\partial w}{\partial V} \cdot B_e \cdot \frac{\partial V}{\partial w} = \frac{\partial g}{\partial w} \ . \tag{1.3.6}$$

Thus, the quasi-linear form of the Euler equations written in terms of conservative variables reads :

$$P_w^{-1} \cdot w_t + A_w \cdot w_x + B_w \cdot w_y = 0 \ , \tag{1.3.7}$$

which eventually leads back to the conservative form :

$$P_w^{-1} \cdot w_t + f^E(w)_x + g^E(w)_y = 0 \ . \tag{1.3.8}$$

Properties of the matrix P_w

The Turkel low-Mach preconditioner written in terms of entropic variables can also be expressed as follows :

$$P_e = Id + (\beta^2 - 1)Q_e \quad \text{with} \quad Q_e = \text{Diag}[1, 0, 0, 0] \tag{1.3.9}$$

where it is worth noticing matrix Q_e is idempotent, *i.e.* $Q_e^2 = Q_e$; consequently the inverse of the preconditioning matrix is simply computed as :

$$P_e^{-1} = Id + (\frac{1}{\beta^2} - 1)Q_e \tag{1.3.10}$$

Since the conservative version of the preconditioner is such that $P_w = \dfrac{\partial w}{\partial V} \cdot P_e \cdot \dfrac{\partial V}{\partial w}$, it follows that :

$$P_w = Id + (\beta^2 - 1)\frac{\partial w}{\partial V} \cdot Q_e \cdot \frac{\partial V}{\partial w} = Id + (\beta^2 - 1)Q_w \tag{1.3.11}$$

and simple calculations yield :

$$Q_w = \frac{\partial w}{\partial V} \cdot Q_e \cdot \frac{\partial V}{\partial w} = \frac{\gamma - 1}{c^2} \left[\begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix} \cdot \begin{pmatrix} \frac{q^2}{2} \; ; & -u \; ; & -v \; ; & 1 \end{pmatrix} \right]. \tag{1.3.12}$$

Obviously, the matrix Q_w is also idempotent ($Q_w^2 = Q_w$) so that $P_w^{-1} = Id + (1/\beta^2 - 1)Q_w$; moreover, the particular form of this matrix Q_w enables to compute the matrix-vector product $Q_w \cdot X$ very easily, as already pointed out in [76] :

$$Q_w \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} = \frac{\gamma - 1}{c^2} \cdot \left(\frac{q^2}{2} X_1 - uX_2 - vX_3 + X_4 \right) \cdot \begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix} \tag{1.3.13}$$

These properties will prove soon very useful when optimizing the cost of an implicit Matrix-Free treatment for all speed flows in Chapter 4. Note that from now on the simple notations P, Q will be used to denote the conservative preconditioner P_w and its associated matrix Q_w.

Preconditioned eigensystem

Pre-multiplying the time-derivative term of the Euler equations by the matrix P^{-1} leads to a new eigensystem since equation (1.3.7) can be re-written as follows :

$$w_t + \tilde{A} \cdot w_x + \tilde{B} \cdot w_y = 0 \tag{1.3.14}$$

where the preconditioned Jacobian matrices $\tilde{A} = P \cdot A$ and $\tilde{B} = P \cdot B$ have been introduced. In order to compute the eigenvectors and eigenvalues of these preconditioned matrices, it is recommended [83, 84] to start from the entropic preconditioned Jacobian matrices $P_e \cdot A_e$, $P_e \cdot B_e$ and to use next relation (1.3.5) to recover the conservative matrices. The eigenvalues of the

preconditioned Jacobian $\tilde{J} = \tilde{A}n_x + \tilde{B}n_y$ (with n_x and n_y the components of the normal vector $\vec{n}$) are given by :

$$\begin{aligned}
\tilde{\lambda}^{(1)} &= un_x + vn_y = V_n \\
\tilde{\lambda}^{(2)} &= V_n \\
\tilde{\lambda}^{(3)} &= \frac{1}{2}(1+\beta^2)V_n + \sqrt{\beta^2 c^2 + \frac{1}{4}(\beta^2-1)^2 V_n^2} \\
\tilde{\lambda}^{(4)} &= \frac{1}{2}(1+\beta^2)V_n - \sqrt{\beta^2 c^2 + \frac{1}{4}(\beta^2-1)^2 V_n^2}
\end{aligned} \tag{1.3.15}$$

In what follows, $\tilde{\Lambda}(n_x, n_y)$ denotes the diagonal matrix Diag$[\tilde{\lambda}^{(i)}]$ containing the preconditioned eigenvalues. Note that when β is of the same order than the Mach number and this latter is small, all the preconditioned eigenvalues are of the same order than the particle velocity V_n. This means the system is no longer ill-conditioned for low speed flows. The preconditioned matrix $P_e \cdot (\tilde{A}_e n_x + \tilde{B}_e n_y)$ can be expressed using $\tilde{\Lambda}$ as follows :

$$P_e \cdot (\tilde{A}_e n_x + \tilde{B}_e n_y) = \tilde{J}_e(n_x, n_y) = \tilde{T}_e(n_x, n_y) \cdot \tilde{\Lambda}(n_x, n_y) \cdot \tilde{T}_e^{-1}(n_x, n_y) \tag{1.3.16}$$

where the eigenvector matrices $\tilde{T}_e$ and $\tilde{T}_e^{-1}$ are given in [84] and reproduced here :

$$\tilde{T}_e = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & -n_y & \dfrac{r\ n_x}{\rho\ \beta^2\ c^2} & \dfrac{s\ n_x}{\rho\ \beta^2\ c^2} \\ 0 & n_x & \dfrac{r\ n_y}{\rho\ \beta^2\ c^2} & \dfrac{s\ n_y}{\rho\ \beta^2\ c^2} \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad \tilde{T}_e^{-1} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & -n_y & n_x & 0 \\ \dfrac{s}{2\ t} & -\dfrac{\rho\ \beta^2\ c^2}{2\ t}\ n_x & -\dfrac{\rho\ \beta^2\ c^2}{2\ t}\ n_y & 0 \\ -\dfrac{r}{2\ t} & \dfrac{\rho\ \beta^2\ c^2}{2\ t}\ n_x & \dfrac{\rho\ \beta^2\ c^2}{2\ t}\ n_y & 0 \end{pmatrix} \tag{1.3.17}$$

with

$$r = \tilde{\lambda}^{(3)} - V_n\beta^2\ , \quad s = \tilde{\lambda}^{(4)} - V_n\beta^2\ , \quad t = \frac{\tilde{\lambda}^{(4)} - \tilde{\lambda}^{(3)}}{2}\ . \tag{1.3.18}$$

In the same manner, the preconditioned Jacobian expressed in terms of conservative variables can also be made diagonal :

$$P \cdot (\tilde{A}n_x + \tilde{B}n_y) = \tilde{J}(n_x, n_y) = \tilde{T}(n_x, n_y) \cdot \tilde{\Lambda}(n_x, n_y) \cdot \tilde{T}^{-1}(n_x, n_y) \tag{1.3.19}$$

Using the transformation relations between entropic and conservative variables, this Jacobian takes also the following form :

$$\tilde{J} = \frac{\partial w}{\partial V} \cdot \tilde{J}_e \cdot \frac{\partial V}{\partial w} = \underbrace{\frac{\partial w}{\partial V} \cdot \tilde{T}_e}_{=\tilde{T}} \cdot \tilde{\Lambda} \cdot \underbrace{\tilde{T}_e^{-1} \cdot \frac{\partial V}{\partial w}}_{=\tilde{T}^{-1}}. \tag{1.3.20}$$

All the above expressions will be used in the next section when building the artificial dissipation of a preconditioned upwind scheme.

1.3.3 Preconditioned upwind scheme

Guillard and Viozat showed in [83, 84, 28] how to build a version of the Roe scheme adapted to the preconditioned system (1.3.8); such a scheme is christened Roe-Turkel scheme in [83] by reference to the use of Turkel preconditioning. This section summarizes the derivation of the Roe-Turkel scheme introduced in [83] with the notations used in the present work. Our starting point is the quasi-linear system (1.3.14), space-discretized using a first-order accurate Roe scheme :

$$w_t + \tilde{A} \cdot w_x + \tilde{B} \cdot w_y = \frac{\delta x}{2}(|\tilde{A}|w_x)_x + \frac{\delta y}{2}(|\tilde{B}|w_y)_y \tag{1.3.21}$$

where it must be understood all the space-derivatives are discretized using simple second-order accurate centered formulas. Left-multiplying this expression by P^{-1} and going back to a conservative form for the flux balance yields the following semi-discrete form for the preconditioned Roe scheme :

$$P^{-1} \cdot w_t + f_x + g_y = \frac{\delta x}{2}(P^{-1}|P \cdot A|w_x)_x + \frac{\delta y}{2}(P^{-1}|P \cdot B|w_y)_y \tag{1.3.22}$$

Guillard and Viozat [28] proved that, for a low Mach number flow, all the components of the above numerical dissipation fluxes remain of the same order of magnitude than the corresponding non-dissipative parts of the numerical fluxes, which enables to preserve accuracy for low speed flows. In order to evaluate this preconditioned dissipation in a simple manner, the different expressions defined in the previous section are now used :

$$P^{-1} \cdot |P \cdot An_x + P \cdot Bn_y| = (P^{-1} \cdot \frac{\partial w}{\partial V} \cdot \tilde{T}_e) \cdot |\tilde{\Lambda}| \cdot (\tilde{T}_e^{-1} \cdot \frac{\partial V}{\partial w}) = \tilde{T}^g \cdot |\tilde{\Lambda}| \cdot \tilde{T}^d \tag{1.3.23}$$

Matrices $\tilde{T}^g$ and $\tilde{T}^d$ are easily computed from matrices $\tilde{T}_e$, $\tilde{T}_e^{-1}$, $\frac{\partial V}{\partial w}$ and P^{-1} since :

$$\begin{aligned} \tilde{T}^g(n_x, n_y) &= P^{-1} \cdot \tilde{T} = \frac{\partial w}{\partial V} \cdot P_e^{-1} \cdot \tilde{T}_e \\ \tilde{T}^d(n_x, n_y) &= \tilde{T}^{-1} = \tilde{T}_e^{-1} \cdot \frac{\partial V}{\partial w} \end{aligned} \tag{1.3.24}$$

and their respective explicit expression reads :

$$\tilde{T}^g = \begin{pmatrix} 1 & 0 & \frac{1}{2\ \beta^2\ c^2} & \frac{1}{2\ \beta^2\ c^2} \\ u & n_y & \frac{u\ +\ r\ n_x}{2\ \beta^2\ c^2} & \frac{u\ +\ s\ n_x}{2\ \beta^2\ c^2} \\ v & -n_x & \frac{v\ +\ r\ n_y}{2\ \beta^2\ c^2} & \frac{v\ +\ s\ n_y}{2\ \beta^2\ c^2} \\ q^2 & -V_t & \frac{H\ +\ r\ V_n}{2\ \beta^2\ c^2} & \frac{H\ +\ s\ V_n}{2\ \beta^2\ c^2} \end{pmatrix} \tag{1.3.25}$$

and

$$\tilde{T}^d = \begin{pmatrix} 1-\frac{\gamma-1}{c^2}q^2 & \frac{\gamma-1}{c^2}u & \frac{\gamma-1}{c^2}v & -\frac{\gamma-1}{c^2} \\ V_t & n_y & -n_x & 0 \\ \frac{sq^2(\gamma-1)+\beta^2c^2V_n}{t} & -\frac{su(\gamma-1)+\beta^2c^2n_x}{t} & -\frac{sv(\gamma-1)+\beta^2c^2n_y}{t} & \frac{s(\gamma-1)}{t} \\ -\frac{rq^2(\gamma-1)+\beta^2c^2V_n}{t} & \frac{ru(\gamma-1)+\beta^2c^2n_x}{t} & \frac{rv(\gamma-1)+\beta^2c^2n_y}{t} & -\frac{r(\gamma-1)}{t} \end{pmatrix} \tag{1.3.26}$$

where $V_t = -un_y + vn_x$, $q^2 = \frac{1}{2}(u^2+v^2)$, $H = E + \frac{p}{\rho}$, $E = e + q^2$ with $p = (\gamma-1)\rho e$ and r, s, t are functions of the preconditioned Jacobian eigenvalues defined by (1.3.18). Note that taking $\beta^2 = 1$ yields $r = c$, $s = -c$ and $t = -c$ so that the matrices given by (1.3.25) and (1.3.26) reduce to the classical matrices which make the Jacobian $(An_x + Bn_y)$ diagonal (see for instance [31]).
The fully-discrete version of the Roe-Turkel scheme (1.3.22) reads :

$$P^{-1}\frac{\Delta w^n_{i,j}}{\Delta t} + (\frac{\delta_1 F}{\delta x} + \frac{\delta_2 G}{\delta y})_{i,j} = 0 \tag{1.3.27}$$

with numerical fluxes defined by :

$$\begin{aligned} F_{i+\frac{1}{2},j} &= (\mu_1 f - \tfrac{1}{2}P^{-1}\cdot|P\cdot A|\cdot\delta_1 w)_{i+\frac{1}{2},j} = (\mu_1 f - \tfrac{1}{2}\tilde{T}^g(1,0)\cdot|\tilde{\Lambda}(1,0)|\cdot\tilde{T}^d(1,0)\cdot\delta_1 w)_{i+\frac{1}{2},j} \\ G_{i,j-\frac{1}{2}} &= (\mu_2 g - \tfrac{1}{2}P^{-1}\cdot|P\cdot B|\cdot\delta_2 w)_{i,j-\frac{1}{2}} = (\mu_2 g - \tfrac{1}{2}\tilde{T}^g(0,1)\cdot|\tilde{\Lambda}(0,1)|\cdot\tilde{T}^d(0,1)\cdot\delta_2 w)_{i,j-\frac{1}{2}} \end{aligned} \tag{1.3.28}$$

Extending the scheme to higher-order accuracy is done by taking :

$$\begin{aligned} F_{i+\frac{1}{2},j} &= \frac{1}{2}(f(w^L_{i+\frac{1}{2},j}) + f(w^R_{i+\frac{1}{2},j})) - \frac{1}{2}P^{-1}_{i+\frac{1}{2},j}\cdot|P_{i+\frac{1}{2},j}\cdot A_{i+\frac{1}{2},j}|\cdot(w^R_{i+\frac{1}{2},j} - w^L_{i+\frac{1}{2},j}) \\ G_{i,j-\frac{1}{2}} &= \frac{1}{2}(g(w^L_{i,j-\frac{1}{2}}) + g(w^R_{i,j-\frac{1}{2}})) - \frac{1}{2}P^{-1}_{i,j-\frac{1}{2}}\cdot|P_{i,j-\frac{1}{2}}\cdot B_{i,j-\frac{1}{2}}|\cdot(w^R_{i,j-\frac{1}{2}} - w^L_{i,j-\frac{1}{2}}) \end{aligned}$$

where the states w^R and w^L are reconstructed using relations defined in section 1.2.2.
If the Rusanov scheme is used instead of the Roe scheme, matrices $|\Lambda(\tilde{1},0)|$, $|\Lambda(\tilde{0},1)|$ are replaced with their respective spectral radius yielding the following numerical fluxes for the preconditioned Rusanov scheme :

$$\begin{aligned} F_{i+\frac{1}{2},j} &= \frac{1}{2}(f(w^L_{i+\frac{1}{2},j}) + f(w^R_{i+\frac{1}{2},j})) - \frac{1}{2}\rho(PA)P^{-1}\cdot(w^R_{i+\frac{1}{2},j} - w^L_{i+\frac{1}{2},j}) \\ G_{i,j-\frac{1}{2}} &= \frac{1}{2}(g(w^L_{i,j-\frac{1}{2}}) + g(w^R_{i,j-\frac{1}{2}})) - \frac{1}{2}\rho(PB)P^{-1}\cdot(w^R_{i,j-\frac{1}{2}} - w^L_{i,j-\frac{1}{2}}) \end{aligned}$$

since $\tilde{T}^g\cdot\tilde{T}^d = P^{-1}$. Note the numerical dissipation of this preconditioned Rusanov scheme is no longer purely scalar since it contains now the preconditioning matrix P^{-1}.
Low-Mach number preconditioned versions of the AUSM+, Harten-Lax-van Leer, Jameson's

CUSP schemes are respectively provided in [46] [51] [77]. In the present work, low-Mach number versions of the Roe, Rusanov and AUSM+ schemes have been used. The Von Neumann analysis carried out in the next chapters has focused on the implicit Roe upwind scheme mainly because its expression is available for the full non-linear Euler equations as well as for its linearized version while, for instance, the derivation of the AUSM+ scheme applied to the linearized form of the Euler equations requires some tedious developments and a number of approximations (see [46] for more details).
For the Roe or Rusanov scheme, the preconditioning method affects in a simple way the definition of the numerical dissipation. From now on, $\tilde{D}_1$ and $\tilde{D}_2$ will denote the preconditioned numerical dissipation coefficients associated with each space direction while the preconditioned numerical inviscid fluxes will be denoted $\tilde{F}^E$ and $\tilde{G}^E$. Thus, the numerical fluxes of the Roe and Rusanov schemes combined with Turkel preconditioning can be unified as :

$$\begin{aligned} \tilde{F}^E_{i+\frac{1}{2},j} &= \frac{1}{2}(f^E(w^L_{i+\frac{1}{2},j}) + f^E(w^R_{i+\frac{1}{2},j})) - \tilde{D}_1 \cdot (w^R_{i+\frac{1}{2},j} - w^L_{i+\frac{1}{2},j}) \\ \tilde{G}^E_{i,j-\frac{1}{2}} &= \frac{1}{2}(g^E(w^L_{i,j-\frac{1}{2}}) + g^E(w^R_{i,j-\frac{1}{2}})) - \tilde{D}_2 \cdot (w^R_{i,j-\frac{1}{2}} - w^L_{i,j-\frac{1}{2}}) \end{aligned} \tag{1.3.29}$$

with

$$\begin{cases} \tilde{D}_1 = \frac{1}{2}P^{-1}|P\,A| = \frac{1}{2}\tilde{T}^g(1,0) \cdot |\Lambda(\tilde{1},0)| \cdot \tilde{T}^d(1,0) \\ \\ \tilde{D}_2 = \frac{1}{2}P^{-1}|P\,B| = \frac{1}{2}\tilde{T}^g(0,1) \cdot |\Lambda(\tilde{0},1)| \cdot \tilde{T}^d(0,1) \end{cases} \tag{1.3.30}$$

for the preconditioned Roe scheme or the Roe-Turkel scheme and

$$\begin{cases} \tilde{D}_1 = \frac{1}{2}\rho(P\ A)P^{-1} \\ \\ \tilde{D}_2 = \frac{1}{2}\rho(P\ B)P^{-1} \end{cases} \tag{1.3.31}$$

for the preconditioned Rusanov scheme. This formulation is extended to the Navier-Stokes system in a straightforward way since the purely centered discretization of the viscous fluxes remains unaffected by the preconditioning of the equations. In other words, the preconditioned Navier-Stoke equations :

$$P^{-1} \cdot w_t \ + \ (f^E(w) - f^V(w, w_x, w_y))_x \ + \ (g^E(w) - g^V(w, w_x, w_y))_y \ = \ 0 \tag{1.3.32}$$

are approximated by the conservative scheme

$$(P^{-1})^n_{i,j} \cdot \frac{\Delta w^n_{i,j}}{\Delta t} + \left(\frac{\delta_1(\tilde{F}^E - F^V)}{\delta x} + \frac{\delta_2(\tilde{G}^E - G^V)}{\delta y} \right)^n_{i,j} = 0 \tag{1.3.33}$$

where the preconditioned inviscid numerical fluxes are given above and the viscous fluxes keep their expression (1.2.34).

Implicit stage

The low-Mach preconditioning technique applied to the Roe or Rusanov scheme modifies the time derivative term and the numerical dissipation with respect to the standard version of these

schemes but introduces no technical difficulty in the process of making scheme (1.3.27) or (1.3.33) implicit; in fact, the form of the implicit preconditioned scheme for the Navier-Stokes equations remains unchanged :

$$\mathcal{H} \cdot \Delta w_{i,j}^n = -\Delta t \cdot \mathcal{R}_{i,j}^n \ , \tag{1.3.34}$$

with

$$\begin{cases} \mathcal{R}_{i,j}^n = \left(\dfrac{\delta_1(\tilde{F}^E - F^V)}{\delta x} + \dfrac{\delta_2(\tilde{G}^E - G^V)}{\delta y} \right)_{i,j}^n \ , \\ \mathcal{H} = \left((P^{-1})^n + \sigma_1 \delta_1 (A^E)^n \mu_1 + \sigma_2 \delta_2 (B^E)^n \mu_2 - \sigma_1 \delta_1 (\tilde{D}_1 + \dfrac{A_1^V}{\delta x})^n \delta_1 - \sigma_2 \delta_2 (\tilde{D}_2 + \dfrac{B_2^V}{\delta y})^n \delta_2 \right) . \end{cases} \tag{1.3.35}$$

The LHS of equation (1.3.34) can be also expanded as :

$$\mathcal{H} \cdot \Delta w_{i,j}^n = C_0 \Delta w_{i,j}^n + C_1^- \Delta w_{i-1,j}^n + C_1^+ \Delta w_{i+1,j}^n + C_2^- \Delta w_{i,j-1}^n + C_2^+ \Delta w_{i,j+1}^n \ , \tag{1.3.36}$$

where the coefficients $C_p^{\pm}$ take now into account the modification of the numerical dissipation terms by the low-Mach preconditioning and C_0 includes the preconditioning matrix :

$$\begin{cases} C_1^- = -\sigma_1 \left(\frac{1}{2} A^E + \tilde{D}_1 + \frac{1}{\delta x} A_1^V \right)_{i-\frac{1}{2},j}^n = -\sigma_1 \left(\mathcal{A}^+ \right)_{i-\frac{1}{2},j}^n \\ C_1^+ = \sigma_1 \left(\frac{1}{2} A^E - \tilde{D}_1 - \frac{1}{\delta x} A_1^V \right)_{i+\frac{1}{2},j}^n = \sigma_1 \left(\mathcal{A}^- \right)_{i+\frac{1}{2},j}^n \\ C_2^- = -\sigma_2 \left(\frac{1}{2} B^E + \tilde{D}_2 + \frac{1}{\delta y} B_2^V \right)_{i,j-\frac{1}{2}}^n = -\sigma_2 \left(\mathcal{B}^+ \right)_{i,j-\frac{1}{2}}^n \\ C_2^+ = \sigma_2 \left(\frac{1}{2} B^E - \tilde{D}_2 - \frac{1}{\delta y} B_2^V \right)_{i,j+\frac{1}{2}}^n = \sigma_2 \left(\mathcal{B}^- \right)_{i,j+\frac{1}{2}}^n \\ C_0 = (P^{-1})_{i,j}^n + \sigma_1 \left(\mathcal{A}^+ \right)_{i+\frac{1}{2},j}^n - \sigma_1 \left(\mathcal{A}^- \right)_{i-\frac{1}{2},j}^n + \sigma_2 \left(\mathcal{B}^+ \right)_{i,j+\frac{1}{2}}^n - \sigma_2 \left(\mathcal{B}^- \right)_{i,j-\frac{1}{2}}^n \end{cases} \tag{1.3.37}$$

When the preconditioned implicit upwind scheme (1.3.34) is applied at each point of the computational domain, the linear block system (1.2.18) is recovered and must be solved in an efficient way.

1.4 Unsteady Flows

1.4.1 Dual-time framework

Unsteady fluid dynamic processes are frequently encountered in engineering problems and are now commonly computed by the CFD practitioner. An immediate way to obtain a time-accurate solution of the (unsteady) Navier-Stokes equations,

$$\frac{\partial w}{\partial t} + (f^E - f^V)_x + (g^E - g^V)_y = 0$$

would be to apply the following scheme :

$$\frac{w_{i,j}^{n+1} - w_{i,j}^{n}}{\Delta t} + \left(\frac{\delta_1 F}{\delta x} + \frac{\delta_2 G}{\delta y} \right)_{i,j}^{n} = 0 \tag{1.4.1}$$

with a unique global time step Δt in each cell of the computational domain. However, such an explicit scheme is only first-order time-accurate which is not sufficient for computing aerodynamic problems, unless very small time-step are used which leads in turn to high CPU time consumption. Furthermore, in the case of low Mach number flow problems, stability criteria would impose further limitations on the time-step size which would impair anyway the usefulness of such an algorithm. Consequently, in order to increase both the accuracy and the efficiency of the time-integration process, it is usual to retain the following implicit scheme :

$$\frac{3w_{i,j}^{n+1} - 4w_{i,j}^{n} + w_{i,j}^{n-1}}{2\Delta t} + \left(\frac{\delta_1 F}{\delta x} + \frac{\delta_2 G}{\delta y} \right)_{i,j}^{n+1} = 0 \tag{1.4.2}$$

An immediate Taylor-development around the new time-level $(n+1)$ shows that scheme (1.4.2) provides a second-order accurate approximation of the unsteady Navier-Stokes equations; moreover, the implicit character of the scheme allows to use large time-steps when advancing in time without encountering stability problems (in practice, the time-step is limited by accuracy issues : too large a time-step prevents from correctly representing the time-evolution of the flow problem). Naturally, it remains to explain how equation (1.4.2) is solved at each time-step in order to yield a new physical state.

Approximate-Newton method

A first way to solve this scheme is to use an iterative approximate-Newton method. State w^{n+1} is replaced by a provisional state $w^{n,k+1}$ and the fluxes are linearized around the iterative level (n,k) which leads to the following iterative implicit scheme :

$$\mathcal{H} \cdot \Delta w_{i,j}^{n,k} = -\Delta t \cdot \mathcal{R}_{i,j}^{n,k} \quad ,$$

with

$$\left\{ \begin{array}{l} \Delta w_{i,j}^{n,k} = w_{i,j}^{n,k+1} - w_{i,j}^{n,k} \ , \\ \\ \mathcal{R}_{i,j}^{n,k} = \dfrac{3w_{i,j}^{n,k} - 4w_{i,j}^{n} + w_{i,j}^{n-1}}{2\Delta t} + \left(\dfrac{\delta_1 F}{\delta x} + \dfrac{\delta_2 G}{\delta y} \right)_{i,j}^{n,k} \ , \\ \\ \mathcal{H} = \left(\dfrac{3}{2} Id + \sigma_1 \delta_1 (A^E) \mu_1 + \sigma_2 \delta_2 (B^E) \mu_2 - \sigma_1 \delta_1 (D_1 + \dfrac{A_1^V}{\delta x}) \delta_1 - \sigma_2 \delta_2 (D_2 + \dfrac{B_2^V}{\delta y}) \delta_2 \right)^{n,k} . \end{array} \right. \tag{1.4.3}$$

At the first sub-iteration, $w^{n,0}$ is set equal to w^n and once the sub-iterative process has converged (*i.e.* $\Delta w^{n,\infty} = 0$), the unsteady equations are clearly satisfied. The solution at the time level $(n+1)$ is then given by $w^{n+1} = w^{n,\infty}$.

Dual-time stepping scheme

An alternative approach for the solution of the unsteady Navier-Stokes equations is the dual-time method which has been commonly used for incompressible flows for a long time now [63] but was first applied to compressible flow computations by Jameson [34]. At each physical time-step, state w^{n+1} is computed as a steady-state with respect to a pseudo-time τ (or dual-time). In practice, an artificial (or dual) time-derivative is introduced in addition to the physical time-derivative leading to the following discretization :

$$\frac{w_{i,j}^{n,m+1} - w_{i,j}^{n}}{\Delta\tau} + \frac{3w_{i,j}^{n,m+1} - 4w_{i,j}^{n} + w_{i,j}^{n-1}}{2\Delta t} + \left(\frac{\delta_1 F}{\delta x} + \frac{\delta_2 G}{\delta y}\right)_{i,j}^{n,m+1} = 0 \tag{1.4.4}$$

where $\Delta\tau$ is the pseudo-time step and $w^{n,m+1}$ denotes the state at pseudo-time $(m+1)\Delta\tau$ in the convergence process from the physical state at time $(n\Delta t)$ to the one at time $(n+1)\Delta t$. Linearizing the numerical fluxes around the pseudo-time level (n,m) yields :

$$\mathcal{H} \cdot \Delta w_{i,j}^{n,m} = -\Delta\tau \cdot \mathcal{R}_{i,j}^{n,m} \quad , \tag{1.4.5}$$

with

$$\begin{cases} \Delta w_{i,j}^{n,m} = w_{i,j}^{n,m+1} - w_{i,j}^{n,m} \ , \\ \\ \mathcal{R}_{i,j}^{n,m} = \dfrac{3w_{i,j}^{n,m} - 4w_{i,j}^{n} + w_{i,j}^{n-1}}{2\Delta t} + \left(\dfrac{\delta_1 F}{\delta x} + \dfrac{\delta_2 G}{\delta y}\right)_{i,j}^{n,m} \ , \\ \\ \mathcal{H} = \left((1+\dfrac{3}{2}\lambda)Id + \sigma_1\delta_1 A^E\mu_1 + \sigma_2\delta_2 B^E\mu_2 - \sigma_1\delta_1(D_1 + \dfrac{A_1^V}{\delta x})\delta_1 - \sigma_2\delta_2(D_2 + \dfrac{B_2^V}{\delta y})\delta_2\right)^{n,m} \ , \end{cases} \tag{1.4.6}$$

where the parameter $\lambda = \frac{\Delta\tau}{\Delta t}$ has been introduced and the quantity σ_1 (resp. σ_2) denotes now the ratio between the pseudo-time step and the space step in the x (resp. y) space-direction : $\sigma_1 = \frac{\Delta\tau}{\delta x}$, $\sigma_2 = \frac{\Delta\tau}{\delta y}$.
As in the approximate-Newton scheme, the first pseudo-state $w^{m+1,0}$ is set equal to w^n, and, when the pseudo-convergence process is achieved, *i.e.* $\Delta w^{n,\infty} = 0$, the LHS of equation (1.4.5) vanishes. The unsteady equations are therefore satisfied (with second-order accuracy in time and a space-accuracy depending on the choice of the numerical fluxes formulas) and the solution at physical-time level $(n+1)$ is given by $w^{n+1} = w^{n,\infty}$. One can notice that taking $\Delta\tau = \infty$ makes the dual-time stepping scheme identical to the approximate-Newton one. In fact, the approximate-Newton method can be seen as a particular case of the dual-time stepping approach. The decisive interest of the dual-time stepping scheme is that it allows a direct re-use of the convergence speedup strategies ordinarily applied for steady flow calculations. Indeed, the LHS of equation (1.4.5) can also be expressed as follows :

$$\mathcal{H} \cdot \Delta w_{i,j}^{n,m} = C_0\Delta w_{i,j}^{n,m} + C_1^-\Delta w_{i-1,j}^{n,m} + C_1^+\Delta w_{i+1,j}^{n,m} + C_2^-\Delta w_{i,j-1}^{n,m} + C_2^+\Delta w_{i,j+1}^{n,m} \ , \tag{1.4.7}$$

where the coefficients $C_i^{\pm}$ are formally unchanged with respect to the steady case (but note the definition of σ_p is now based on the dual time-step) and C_0 is slightly modified since it includes

now the ratio λ between the physical and dual time-steps :

$$\begin{cases} C_1^- = -\sigma_1 \left(\frac{1}{2}A^E + D_1 + \frac{1}{\delta x}A_1^V\right)_{i-\frac{1}{2},j}^{n,m} = -\sigma_1 \left(\mathcal{A}^+\right)_{i-\frac{1}{2},j}^{n,m} \\ C_1^+ = \sigma_1 \left(\frac{1}{2}A^E - D_1 - \frac{1}{\delta x}A_1^V\right)_{i+\frac{1}{2},j}^{n,m} = \sigma_1 \left(\mathcal{A}^-\right)_{i+\frac{1}{2},j}^{n,m} \\ C_2^- = -\sigma_2 \left(\frac{1}{2}B^E + D_2 + \frac{1}{\delta y}B_2^V\right)_{i,j-\frac{1}{2}}^{n,m} = -\sigma_2 \left(\mathcal{B}^+\right)_{i,j-\frac{1}{2}}^{n,m} \\ C_2^+ = \sigma_2 \left(\frac{1}{2}B^E - D_2 - \frac{1}{\delta y}B_2^V\right)_{i,j+\frac{1}{2}}^{n,m} = \sigma_2 \left(\mathcal{B}^-\right)_{i,j+\frac{1}{2}}^{n,m} \\ C_0 = (1+\frac{3}{2}\lambda)Id + \sigma_1 \left(\mathcal{A}^+\right)_{i+\frac{1}{2},j}^{n,m} - \sigma_1 \left(\mathcal{A}^-\right)_{i-\frac{1}{2},j}^{n,m} + \sigma_2 \left(\mathcal{B}^+\right)_{i,j+\frac{1}{2}}^{n,m} - \sigma_2 \left(\mathcal{B}^-\right)_{i,j-\frac{1}{2}}^{n,m} \end{cases} \tag{1.4.8}$$

Applying the time-accurate implicit upwind scheme (1.4.5) at each point of the computational domain, one obtains a linear block system similar to the one expressed in (1.2.18). Hence, one can use the same treatments employed for steady cases in order to solve the time-dependent system.

1.4.2 Unsteady low-Mach number flows

The dual-time stepping formulation makes the implementation of the low Mach preconditioning very easy. Indeed, applying the preconditioning matrix to the pseudo-time derivative enables to preserve the time-consistency while the convergence rate of the dual-process is improved and the accuracy of the computation is ensured by the preconditioned numerical dissipation. Thus the form of the time-accurate implicit scheme remains the same :

$$\mathcal{H} \cdot \Delta w_{i,j}^{n,m} = -\Delta\tau \cdot \mathcal{R}_{i,j}^{n,m} \quad , \tag{1.4.9}$$

with

$$\begin{cases} \Delta w_{i,j}^{n,m} = w_{i,j}^{n,m+1} - w_{i,j}^{n,m} \ , \\ \mathcal{R}_{i,j}^{n,m} = \dfrac{3w_{i,j}^{n,m} - 4w_{i,j}^{n} + w_{i,j}^{n-1}}{2\Delta t} + \left(\dfrac{\delta_1(\tilde{F}^E - F^V)}{\delta x} + \dfrac{\delta_2(\tilde{G}^E - G^V)}{\delta y}\right)_{i,j}^{n,m} \ , \\ \mathcal{H} = \left(P^{-1} + \dfrac{3}{2}\lambda Id + \sigma_1\delta_1 A^E\mu_1 + \sigma_2\delta_2 B^E\mu_2 - \sigma_1\delta_1(\tilde{D}_1 + \dfrac{A_1^V}{\delta x})\delta_1 - \sigma_2\delta_2(\tilde{D}_2 + \dfrac{B_2^V}{\delta y})\delta_2\right)^{n,m} \ , \end{cases} \tag{1.4.10}$$

where we recall that $\tilde{F}^E$ and $\tilde{G}^E$ are the preconditioned convective fluxes (from the Roe-Turkel scheme for instance) while $\tilde{D}_1$ and $\tilde{D}_2$ are the preconditioned numerical dissipation terms.
As mentioned earlier in the section dedicated to low Mach preconditioning, the efficiency and the robustness of the preconditioning method for the unsteady equations rely on the value of the "cut-off" used in the definition of the β parameter. Venkateswaran *et al.* showed in [81] that it is necessary to modify the definition of the "cut-off" when going from steady to unsteady flow problems. Using Von Neumann analysis, they proved that steady preconditioning is particularly inefficient for computing unsteady flows. In the chapter dedicated to the Von Neumann analysis, we will resume their work in the case of the implicit Matrix-Free method.

1.5 Conclusions

This introductory chapter has recalled the design principles for building a block implicit upwind scheme, successively for the steady compressible Euler and Navier-Stokes equations, then their low-Mach number (preconditioned) version and finally in the unsteady low-Mach number case treated using a dual time-step technique. Going from the steady compressible Euler equations to the unsteady low-Mach number Navier-Stokes equations does not modify formally the linear block system that must be solved at each iteration in order to obtain a new value for the vector increment of conserved variables Δw. If we restrict ourselves to a three-point per direction first-order linearization of the inviscid numerical fluxes, the block implicit scheme reads in all the cases :

$$\mathcal{H} \cdot \Delta w_{i,j}^{n,m} = -\Delta\tau \cdot \mathcal{R}_{i,j}^{n,m} \quad ,$$

with an implicit operator of the form

$$\mathcal{H} \cdot \Delta w_{i,j}^{n,m} = C_0 \Delta w_{i,j}^{n,m} + C_1^- \Delta w_{i-1,j}^{n,m} + C_1^+ \Delta w_{i+1,j}^{n,m} + C_2^- \Delta w_{i,j-1}^{n,m} + C_2^+ \Delta w_{i,j+1}^{n,m} \ ,$$

where $C_p^{\pm}$, C_0 are 4×4 (in 2D) block matrices built from the inviscid and viscous Jacobian matrices plus the preconditioning matrix for low-Mach number flows and the λ parameter for dual-time computation. It will be shown in chapter 3 that the intrinsic efficiency of such a block implicit stage is quite high; however, solving directly (with a Gauss-inversion) the linear system associated with the implicit stage would become much too expensive as soon as the grid size increases. In practice, this linear system is approximately solved and different solution methods can be considered to this end. The next chapter describes two main classes of standard solution methods : approximate factorization techniques and relaxation techniques, with a view to select the one providing the best efficiency.

Chapter 2

Implicit Treatments

The block implicit stages described in the previous chapter can be ideally solved as such, meaning the linear system associated with the implicit stage can be exactly solved (using some Gauss-type inversion method for instance) but this is very rarely done in practice because such a direct solver would be much too computational time and memory consuming ! In what follows however, the direct solution of a given implicit stage will be considered as a reference as far as the intrinsic efficiency is concerned since such a direct solution does provide the lowest number of iterations to reach a steady-state (though, once again, at the expense of an unacceptably large unit cost). There exists an ongoing quest for efficient approximate solutions of implicit stages, as proved for instance by a recent contribution of Jameson and Caughey [36] : the basic motivation of this quest has first been to find techniques that would simply allow to approximately solve the implicit stage for a reasonable unit cost and the main achievement in that respect has been the development of approximate factorization methods, with decisive contributions in the field of Computational Fluid Dynamics by Beam and Warming [3] and Briley and MacDonald [8]; approximate factorization methods, in spite of some limitations that will be recalled in the next chapter when studying the amplification factor of various implicit treatments, are still much in favor in some heavily used flow solvers (see [62] for instance where an approximate factorization technique is used within the OVERFLOW code; note that in this specific case the implicit stage is made diagonal per direction, taking advantage of the specific form of approximate factorization, which makes the approach attractive for its reduced unit cost and lowered memory requirement). Next, with the development of iterative techniques, the focus has switched on finding an approximate (iterative) solution method allowing to achieve an intrinsic efficiency similar to that provided by the direct solver for the lowest possible unit cost. Following the pioneering work of MacCormack [52] [57], numerous applications of relaxation techniques have been successfully applied to implicit upwind schemes. For several years, researchers and engineers working on the development of efficient implicit schemes have divided themselves between those favoring the approximate factorization methods and the ones promoting the use of relaxation techniques. However, with the development of iterative approximation methods, (re)introduced in the late nineties in particular by MacCormack [54] [55] [56], MacCormack, Pulliam et Venkateswaran [64], it turned out such a division was largely artificial : line-relaxation and approximate factorization techniques are now viewed as belonging to the same family of iterative solution methods, with the standard ADI or modified ADI technique a particular case where a single iteration is performed [9], [22]. For the sake of clarity, the presentation of these various techniques in the present chapter will be progressive, starting with the basic factorization approach, followed

by its more recent iterative version and ending up with point or line-relaxation approach; the connection between factorization and relaxation techniques will be underlined throughout this chapter as well as in the next one, devoted to the Von Neumann analysis of the implicit solution methods here reviewed.

2.1 Approximate Factorization Methods

As previously mentioned, approximate factorization methods were the first successful implicit methods used in CFD. Before the mid-1980s and the development of upwind implicit schemes, the use of central difference schemes implied a lack of diagonal dominance of the matrix M of the linear system (1.2.18) which made the standard relaxation techniques unstable. That is why factorization techniques met a great success until recent years. In this section, two widely-used approximate factorization methods are presented: the alternate direction implicit method (ADI) and the modified approximate factorization method (MAF) [54] also sometimes referred to as Dominant Diagonal ADI or DDADI [64].

2.1.1 Alternating Direction Implicit Method

The alternating direction implicit method (ADI), also called approximate factorization method (AF), was successfully used in CFD for the first time by Beam and Warming [3] and Briley and MacDonald [8]. For standard compressible flows, the matrix M of the linear system (1.2.18) is approached by the product of two tri-diagonal matrices, but applying this method to low-Mach number flows requires an additional diagonal matrix so that the factorization is carried out as follows :

$$M \approx M_1 \cdot D^{-1} \cdot M_2 \tag{2.1.1}$$

with

$$M_1 = \begin{bmatrix} x & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ x & x & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & x & x & \cdot & \cdot & \cdot \\ \cdot & \cdot & C_1^- & C_1^0 & C_1^+ & \cdot & \cdot \\ \cdot & \cdot & \cdot & x & x & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & x & x \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & x \end{bmatrix}, \quad M_2 = \begin{bmatrix} x & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & \cdot & x & \cdot & \cdot & \cdot \\ x & \cdot & x & \cdot & x & \cdot & \cdot \\ \cdot & C_2^- & \cdot & C_2^0 & \cdot & C_2^+ & \cdot \\ \cdot & \cdot & x & \cdot & x & \cdot & x \\ \cdot & \cdot & \cdot & x & \cdot & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & x \end{bmatrix}, \tag{2.1.2}$$

and

$$D = \begin{bmatrix} x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & C_0^0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x \end{bmatrix}, \tag{2.1.3}$$

where the diagonal coefficients C_1^0, C_2^0 and C_0^0 are defined by :

$$\begin{cases} C_1^0 = P^{-1} + \sigma_1 \left(\frac{1}{2}A^E + \tilde{D}_1\right)_{i+\frac{1}{2},j} - \sigma_1 \left(\frac{1}{2}A^E - \tilde{D}_1\right)_{i-\frac{1}{2},j} \quad , \\ C_2^0 = P^{-1} + \sigma_2 \left(\frac{1}{2}B^E + \tilde{D}_2\right)_{i,j+\frac{1}{2}} - \sigma_2 \left(\frac{1}{2}B^E - \tilde{D}_2\right)_{i,j-\frac{1}{2}} \quad , \\ C_0^0 = P^{-1} \quad . \end{cases}$$

One can notice that for compressible flows the preconditioning matrix P becomes equal to the identity matrix so that one recovers the standard ADI factorization. Once the factorization performed, the linear system can be solved in three steps as follows :

$$\begin{aligned} &M_1 \cdot [\Delta w^*] = [\mathcal{R}^n] \quad , \\ &[\Delta w^{**}] = D \cdot [\Delta w^*] \quad , \\ &M_2 \cdot [\Delta w^n] = [\Delta w^{**}] \quad , \end{aligned} \tag{2.1.4}$$

where Δw^* and Δw^{**} are provisional values stored respectively after the first and the second step. For carrying out the first and third steps, all is needed is the inversion of a tri-diagonal matrix, which can be easily - viz. at a reduced cost - achieved using a LU factorization. Nevertheless, the ADI method produces a factorization error which can limit the stability of the method in some cases and most of all adversely affects the intrinsic efficiency of this method. The stability and convergence properties of the ADI technique are now well established and understood (see for instance [64]); in the next chapter, we will summarize, through a Von Neumann analysis, the main weakness of this approach, namely its lack of intrinsic efficiency. Let us recall however the ADI treatment applied to a block implicit stage can be made diagonal per direction [65] which allows to significantly reduce its unit cost.

2.1.2 Modified Approximate Factorization Method

The standard ADI method can be modified in order to reduce the factorization error and thus improve its intrinsic efficiency properties, in particular when large time-steps or CFL numbers are used [56]. This can be achieved by approximating the matrix M associated with the implicit stage as follows :

$$M \approx M_1' \cdot D^{-1} \cdot M_2' \tag{2.1.5}$$

with

$$M_1' = \begin{bmatrix} x & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ x & x & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & x & x & \cdot & \cdot & \cdot \\ \cdot & \cdot & C_1^- & C_0 & C_1^+ & \cdot & \cdot \\ \cdot & \cdot & \cdot & x & x & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & x & x \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & x \end{bmatrix} , \quad M_2' = \begin{bmatrix} x & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & \cdot & x & \cdot & \cdot & \cdot \\ x & \cdot & x & \cdot & x & \cdot & \cdot \\ \cdot & C_2^- & \cdot & C_0 & \cdot & C_2^+ & \cdot \\ \cdot & \cdot & x & \cdot & x & \cdot & x \\ \cdot & \cdot & \cdot & x & \cdot & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & x \end{bmatrix} ,$$

$$D = \begin{bmatrix} x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & C_0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x \end{bmatrix} . \tag{2.1.6}$$

Thanks to this modified factorization, non-zero coefficients of the matrix M are left unchanged, while initially zero coefficients can still be modified which means a factorization error is still introduced - it is however much reduced with respect to the baseline ADI approach, in particular when large time-steps are used. The modified approximate factorization (MAF) is performed in three steps :

$$\begin{aligned} &M_1' \cdot [\Delta w^*] = [\mathcal{R}^n] \quad , \\ &[\Delta w^{**}] = D \cdot [\Delta w^*] \quad , \\ &M_2' \cdot [\Delta w^n] = [\Delta w^{**}] \quad . \end{aligned} \tag{2.1.7}$$

The MAF method is also called DDADI method to emphasize the fact that the matrices involved in the factorization process are diagonal dominant [12]. The Von Neumann analysis presented in the next chapter will demonstrate the MAF approach provides a better convergence rate than the basic AF or ADI approach, though still far from the intrinsic efficiency associated with a direct solution of the implicit stage.

2.1.3 Iterative correction of factorization error

ADI and MAF methods introduce a factorization error which can be expressed as follows :

$$\begin{aligned} &M_1 \cdot D^{-1} \cdot M_2 = M + P_{ADI} \quad , \\ &M_1' \cdot D^{-1} \cdot M_2' = M + P_{MAF} \quad . \end{aligned}$$

In other terms, the actual system solved when using a factorization method is the following one :

$$M \cdot [\Delta w^n] + P \cdot [\Delta w^n] = [\mathcal{R}^n] \quad .$$

In order to eliminate this factorization error, a sub-iterative process can be employed :

$$\begin{aligned}
&[\Delta w^{(0)}] = 0 \\
&l = 0, p \\
&M \cdot [\Delta w^{(l+1)}] + P \cdot [\Delta w^{(l+1)}] = [\mathcal{R}^n] + P \cdot [\Delta w^{(l)}] \\
&[\Delta w^n] = [\Delta w^{(p)}]
\end{aligned} \tag{2.1.8}$$

where $\Delta w^{(l)} = w^{(l)} - w^n$. When the method has converged, *i.e.* when $w^{(l+1)} = w^{(l)}$, the factorization error vanishes and the exact solution of the linear system (1.2.18) is obtained. In practice, the previous algorithm is written in a different way, so as to avoid the expensive evaluation of the factorization error P; for instance the iterative MAF algorithm is rather expressed as :

$$\begin{aligned}
&[\Delta w^{(0)}] = 0 \\
&l = 0, p \\
&M_1' \cdot D^{-1} \cdot M_2' \cdot [w^{(l+1)} - w^{(l)}] = [\mathcal{R}^n] - M \cdot [w^{(l)} - w^n] \\
&[\Delta w^n] = [w^{(p)} - w^n]
\end{aligned} \tag{2.1.9}$$

or even more explicitly :

$$\begin{aligned}
&[\Delta w^{(0)}] = 0 \\
&l = 0, p \\
&M_1' \cdot [w^* - w^{(l)}] = [\mathcal{R}^n] - M \cdot [w^{(l)} - w^n] \\
&[w^{**} - w^*] = D \cdot [w^* - w^{(l)}] \\
&M_2' \cdot [w^{(l+1)} - w^{(l)}] = [w^{**} - w^*] \\
&[\Delta w^n] = [w^{(p)} - w^n]
\end{aligned} \tag{2.1.10}$$

The iterative MAF method requires the same tri-diagonal matrix inversions than the standard MAF technique. Moreover these inversions can be done just once at the beginning of the sub-iterative process. The over-cost of the iterative method is only due to the computation of the RHS at each sub-iteration, which remains very cheap. The convergence rate improvement provided by MAF will be assessed through a Von Neumann analysis in the next chapter.

2.2 Relaxation Methods

In the mid-eighties, MacCormack proposed to use relaxation techniques in order to increase the efficiency of implicit treatments [52, 53]. Prior to that period, the implicit LHS operator was generally inverted using the previously presented AF or ADI factorization method. The use of relaxation methods was promoted by the simultaneous development of implicit *upwind* schemes which ensured the diagonal dominance of the system - even though some studies have proved block-implicit centered schemes of Lax-Wendroff type could also be efficiently solved using those same line-relaxation techniques [19] [16]. Since that time, many studies have been dedicated to these methods, and, more recently, Venkateswaran *et al.* [12] carried out a wide study to analyze the performance capabilities of implicit upwind schemes solved by these techniques [9] [22].
In this section we introduce the principles of the relaxation methods and then we deal with the most popular of them.

2.2.1 Principles and properties

As always, our starting point is the linear system associated with the implicit upwind scheme described in the previous chapter :

$$M \cdot [\Delta w^n] = [\mathcal{R}^n] \quad ,$$

A generic relaxation technique may be expressed by the following step :

$$M' \cdot [\Delta w^{(l+1)}] = [\mathcal{R}^n] - (M - M') \cdot [\Delta w^{(l)}] \quad , \tag{2.2.1}$$

where $\Delta w^{(l)} = w^{(l)} - w^n$ and M' derives from a particular splitting of the matrix M; more precisely, M' is chosen so as to ensure the above linear system will be easier to solve than the initial one : typically, M' is chosen as a tridiagonal matrix since a tridiagonal system can be solved for a reduced cost using a particular LU factorization known as Thomas algorithm [31] [47]. In order to increase the intrinsic efficiency of relaxation methods, it is customary to split M in different ways, corresponding for instance to alternate line or / and symmetric sweeps; for instance, a relaxation technique making use of two sweeps would read :

$$\begin{aligned} M'_1 \cdot [\Delta w^*] &= [\mathcal{R}^n] - (M - M'_1) \cdot [\Delta w^{(l)}] \quad , \\ M'_2 \cdot [\Delta w^{(l+1)}] &= [\mathcal{R}^n] - (M - M'_2) \cdot [\Delta w^*] \quad , \end{aligned} \tag{2.2.2}$$

where Δw^* is a provisional value after the first relaxation sweep. Setting $M'' = M'_1 + M'_2 - M$, these two steps can be combined and written as :

$$M'_1 \cdot M''^{-1} \cdot M'_2 \cdot [w^{(l+1)} - w^{(l)}] = [\mathcal{R}^n] - M \cdot [w^{(l)} - w^n] \quad , \tag{2.2.3}$$

which is an iterative approximate factorization version of the linear system. Expression (2.2.3) makes clear the close connection between relaxation methods and iterative approximate factorization techniques, as shown in [12]. In equation (2.2.3), l represents a "sub" iteration counter which corresponds to multiple sweeps of the linear system, while n denotes the non-linear counter of the time-marching procedure. In practice, $w^{(0)}$ is initialized as $w^{(0)} = w^n$ and p sub-iterations are carried out. When the method has converged, $w^{(l+1)} = w^{(l)}$ and the RHS of equation (2.2.3) is satisfied; w^{n+1} is then set equal to $w^{(p)}$ and the solution is advanced to the next time step. The efficiency of the relaxation method relies on the number of the sub-iterations that are necessary to approach the exact solution of the linear system. If p is too small, the method would be fast - *i.e.* the unit cost would be low - but a great number of non-linear iterations would be necessary to reach the steady state, whereas if p is taken too large, the unit cost method would be higher but it would take fewer non-linear iterations to converge to steady-state. Thus the choice of p is crucial for determining the intrinsic efficiency of the relaxation method. However, this choice may turn to be case-dependent as will be seen in the following chapters, so that it may be difficult to know *a priori* the optimum value of p.
Let us now review some of the most popular relaxation techniques. Each of them corresponds to a specific choice of matrices M'_1 and M'_2 in (2.2.2) with corresponding specific properties of stability and efficiency. Before describing in detail these methods, let us introduce some matrices that will be helpful in the rest of the section; the matrix M associated with the implicit stage may be decomposed as follows :

$$M = D + L_1 + U_1 + L_2 + U_2$$

where the lower and upper matrices associated with each space direction are given by :

$$L_1 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & C_1^- & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot \end{bmatrix} \quad , \quad U_1 = \begin{bmatrix} \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & C_1^+ & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \quad ,$$

$$L_2 = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & C_2^- & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot \end{bmatrix} \quad , \quad U_2 = \begin{bmatrix} \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & C_2^+ & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \quad . \tag{2.2.4}$$

2.2.2 Alternate-Line Jacobi relaxation procedure

This well-known relaxation method relies on the following splitting matrices :

$$\begin{aligned} M_1' &= L_1 + D + U_1 \quad , \\ M_2' &= L_2 + D + U_2 \quad , \end{aligned} \tag{2.2.5}$$

which correspond respectively to the relaxation of extra-diagonal unknown increments Δw in the y direction when a system is solved with M_1' as LHS matrix and to the relaxation of extra-diagonal unknown increments Δw in the x direction when the system is solved with M_2'. The alternate-line Jacobi algorithm, denote hereafter $ALJ(p)$, is expressed as follows :

$$\begin{aligned} &[\Delta w^{(0)}] = 0 \\ &l = 0, p \\ &M_1' \cdot [\Delta w^*] = [\mathcal{R}^n] - (L_2 + U_2) \cdot [\Delta w^{(l)}] \\ &M_2' \cdot [\Delta w^{(l+1)}] = [\mathcal{R}^n] - (L_1 + U_1) \cdot [\Delta w^*] \\ &[\Delta w^n] = [\Delta w^{(p)}] \end{aligned} \tag{2.2.6}$$

The first step corresponds to a line-implicit x-sweep, while the second step corresponds to a line-implicit y-sweep. The cost per sub-iteration of this method is not so expensive since one simply needs to invert a simple tridiagonal block system in each space direction, which can be easily achieved by the use of Thomas algorithm and has to performed only once per non-linear iteration : as in the case of iterative error correction for approximate factorization, only the RHS of (2.2.6) have to be recomputed at each l sub-iteration. However, the global cost of the method will remain low only if the total number p of sub-iterations is not too large. This point will be carefully analyzed in chapter 3. It can already be stated that $ALJ(p)$ is generally not optimal in terms of global efficiency : relaxation methods of Gauss-Seidel type, that make use of the states already evaluated within a sweep, are in general more efficient, as will be shown in the next chapter through the Von Neumann analysis of the amplification factor.

2.2.3 Alternate-Line Gauss-Seidel relaxation procedure

The alternate-line Gauss-Seidel method performs symmetric x-sweeps followed by symmetric y-sweeps. During each sweep, the states which have already been evaluated and whose new values are available are used to compute the next states. This leads to the alternate-line Gauss-Seidel algorithm (ALGS(p)) of the form :

$$\begin{aligned}
&[\Delta w^{(0)}] = 0 \\
&l = 0, p \\
&(M_1' + L_2) \cdot [\Delta w^*] = [\mathcal{R}^n] - U_2 \cdot [\Delta w^{(l)}] \\
&(M_1' + U_2) \cdot [\Delta w^{**}] = [\mathcal{R}^n] - L_2 \cdot [\Delta w^*] \\
&(M_2' + L_1) \cdot [\Delta w^{***}] = [\mathcal{R}^n] - U_1 \cdot [\Delta w^{**}] \\
&(M_2' + U_1) \cdot [\Delta w^{(l+1)}] = [\mathcal{R}^n] - L_1 \cdot [\Delta w^{***}] \\
&[\Delta w^n] = [\Delta w^{(p)}]
\end{aligned} \tag{2.2.7}$$

As will be shown in the next chapter, this ALGS algorithm provides a better convergence rate than the ALJ algorithm. However, as already observed in [12], it will also be shown that the above procedure is only conditionally stable - which may seem rather surprising since $ALGS(p)$ is a widely-used line-relaxation procedure but we will comment on this point later on. The $ALGS(p)$ method allows to recover the level of intrinsic efficiency provided by a direct solution of the implicit stage for a very small number p of sub-iterations; the price to pay for such a high intrinsic efficiency is the need to invert 4 tridiagonal systems in (2.2.7) at each non-linear iteration. An alternative strategy would be to use a relaxation method that does not require any system inversion, at the expense of a probably much lower intrinsic efficiency, even for large values of p. The balance between this poor intrinsic efficiency and the particularly low unit cost of the approach may lead to a globally efficient strategy.

2.2.4 Point Jacobi relaxation procedure

The Point Jacobi or PJ relaxation method is precisely the simplest conceivable relaxation technique since it retains only diagonal terms at the new level $(l+1)$ in the LHS; the following choice for matrix M' :

$$M' = D \quad , \tag{2.2.8}$$

leads to the point Jacobi algorithm (PJ(p)) :

$$\begin{aligned}
&[\Delta w^{(0)}] = 0 \\
&l = 0, p \\
&D \cdot [\Delta w^{(l+1)}] = [\mathcal{R}^n] - (L_2 + L_1 + U_1 + U_2) \cdot [\Delta w^{(l)}] \\
&[\Delta w^n] = [\Delta w^{(p)}]
\end{aligned} \tag{2.2.9}$$

Naturally, the point Jacobi algorithm is expected to have a much poorer convergence rate than $ALGS(p)$: the quantification of this intrinsic efficiency loss will be performed in the next chapter through Fourier analysis. Moreover, Von Neumann analysis will reveal that $PJ(p)$ is in fact unstable when it is used in conjunction with a block implicit scheme. Nevertheless, this algorithm will prove to be a very cheap method to solve the matrix-free implicit scheme.

2.2.5 Symmetric Point Gauss-Seidel relaxation procedure

The Point Jacobi algorithm can also be improved using a Gauss-Seidel type procedure : states evaluated during a sweep are immediately used to compute the next ones; moreover a reverse sweep is performed to increase the efficiency of the method. Consequently, the choice of the matrices associated with such a procedure is the following one :

$$\begin{aligned} M_1' &= L_2 + L_1 + D \quad , \\ M_2' &= D + U_1 + U_2 \quad , \end{aligned} \tag{2.2.10}$$

which leads to the symmetric point Gauss-Seidel algorithm (SGS(p)) :

$$\begin{aligned} &[\Delta w^{(0)}] = 0 \\ &l = 0, p \\ &M_1' \cdot [\Delta w^*] = [\mathcal{R}^n] - (U_1 + U_2) \cdot [\Delta w^{(l)}] \\ &M_2' \cdot [\Delta w^{(l+1)}] = [\mathcal{R}^n] - (L_1 + L_2) \cdot [\Delta w^*] \\ &[\Delta w^n] = [\Delta w^{(p)}] \end{aligned} \tag{2.2.11}$$

It must be well understood that the above algorithm does not introduce any actual inversion of tridiagonal systems ! It could also be written as :

$$\begin{aligned} &[\Delta w^{(0)}] = 0 \\ &l = 0, p \\ &D \cdot [\Delta w^*] = [\mathcal{R}^n] - (L_1 + L_2) \cdot [\Delta w^*] - (U_1 + U_2) \cdot [\Delta w^{(l)}] \\ &D \cdot [\Delta w^{(l+1)}] = [\mathcal{R}^n] - (U_1 + U_2) \cdot [\Delta w^{(l+1)}] - (L_1 + L_2) \cdot [\Delta w^*] \\ &[\Delta w^n] = [\Delta w^{(p)}] \end{aligned} \tag{2.2.12}$$

in order to make clearer the fact that the quantity $(L_1 + L_2) \cdot [\Delta w^*]$ in the first sweep can be computed using the new states Δw^* by simply taking advantage of immediate actualization within the sweep; the same comment holds concerning the quantity $(U_1 + U_2) \cdot [\Delta w^{(l+1)}]$ in the second sweep.
Since the first matrix used in 2.2.11 is a lower triangular matrix and the second one is an upper triangular matrix, this algorithm is sometimes referred to as LU-SGS. Using expression (2.2.3), the above SGS relaxation procedure also reads :

$$M_1' \cdot D^{-1} \cdot M_2' \cdot [w^{(l+1)} - w^{(l)}] = [\mathcal{R}^n] - M \cdot [w^{(l)} - w^n]$$

with an associated algorithm that can be expressed as :

$$\begin{aligned} &[\Delta w^{(0)}] = 0 \\ &l = 0, p \\ &M_1' \cdot [w^* - w^{(l)}] = [\mathcal{R}^n] - M \cdot [w^{(l)} - w^n] \\ &[w^{**} - w^*] = D \cdot [w^* - w^{(l)}] \\ &M_2' \cdot [w^{(l+1)} - w^{(l)}] = [w^{**} - w^*] \\ &[\Delta w^n] = [w^{(p)} - w^n] \end{aligned} \tag{2.2.13}$$

Clearly, when a single sub-iteration is performed, the well-known LU approximate factorization method is recovered. This remarks is aimed at pointing out once again the strong link between factorization methods and relaxation procedures. As will be shown in the next chapter on the basis of a Von Neumann analysis, the SGS algorithm allows to dramatically improve the convergence rate with respect to the basic PJ method.

2.3 Conclusions

Several implicit treatments have been reviewed, that can be used to approximate at each time-step the solution of the block implicit scheme presented in the first chapter of this thesis. Dividing them between factorization and relaxation techniques would be rather artificial since standard factorization techniques are nothing but iterative factorization technique applied with a single iteration; it seems more interesting to distinguish between those treatments that make use of tridiagonal systems inversions (AF(p), MAF(p), ALJ(p), ALGS(p)) and those that do not require any system inversion (PJ(p), LU-SGS(p)). The latter treatments are bound to provide a rather poor intrinsic efficiency but are nonetheless interesting for their low unit cost whereas the former treatments are naturally more time-consuming but offer a much better convergence rate. Such qualitative comments are naturally not sufficient to allow the selection of the best globally efficient block implicit treatment hence the quantitative Von Neumann analysis performed in the next chapter.

Chapter 3

Intrinsic Efficiency of Implicit Block Schemes

3.1 Introduction and objectives

Most of the analytical tools used to study the convergence performance of numerical schemes are limited to linear problems. However, performing a stability analysis is not easy even with such a simplification. Among the various methods available to carry out this task, Von Neumann analysis is certainly the most valuable. It is based on the Fourier transform of the numerical scheme and it allows to know whether the numerical errors introduced in the solution by the discretization are amplified (instability) or damped (stability) [31] [47]. Besides, Von Neumann analysis can be used to estimate how fast the error will be damped (or amplified), providing a quantitative information to rank a variety of implicit treatments on the basis of their convergence rate. Of course, the results obtained by such a method have to be commented carefully. Indeed, as mention upper, Von Neumann theory requires a linear problem which means that the coefficients of the equations are constant; moreover, periodic boundary conditions are assumed in order to allow the use of Fourier series. In spite of these limitations, many researchers, such as Merkle's co-workers in the USA [14, 11, 81] and Lerat's team in France [39, 43, 44], have employed this technique to study the stability and efficiency properties of numerical schemes not only applied to linear scalar equations but also when used to solve the full (linearized) Euler or Navier-Stokes. The stability (and efficiency) results obtained in this latter case are in general - and in spite of the above-mentioned restrictions on the scope of Von Neumann analysis - meaningful for practical computations. Therefore, this chapter will be focused on the Von Neumann analysis of the implicit treatments presented in chapter 2, when used to solve the block-implicit stage of the Roe upwind scheme presented in the first chapter of the thesis in the case of the linearized Euler equations. The goal of this preliminary study is to identify the numerical treatment providing the best global efficiency for all-speed flows, regardless of memory-storage issue for the time being. The select block-implicit treatment will stand as a reference point for the matrix-free implicit treatment introduced in the next chapter. In the present chapter, the principles of the Von Neumann analysis are first briefly outlined; this analysis is then applied to the block-implicit Roe-type upwind scheme and global and local stability / efficiency results are presented both for the reference direct solver and the iterative methods introduced in the previous chapter.

3.2 Linearization of the implicit upwind scheme

Let us consider the implicit upwind scheme for low-Mach unsteady flows (1.4.9) that was described at the end of the first chapter. All other cases (steady inviscid flows and so on), also described in chapter 1, can be derived from this scheme after different simplifications. This scheme reads, using the notations introduced in chapter 1 :

$$\mathcal{H} \cdot \Delta w_{i,j}^{n,m} = -\Delta\tau \cdot \mathcal{R}_{i,j}^{n,m} \quad ,$$

with

$$\left\{ \begin{array}{l} \Delta w_{i,j}^{n,m} = w_{i,j}^{n,m+1} - w_{i,j}^{n,m} \ , \\ \\ \mathcal{R}_{i,j}^{n,m} = \dfrac{3w_{i,j}^{n,m} - 4w_{i,j}^{n} + w_{i,j}^{n-1}}{2\Delta t} + \left(\dfrac{\delta_1(\tilde{F}^E - F^V)}{\delta x} + \dfrac{\delta_2(\tilde{G}^E - G^V)}{\delta y} \right)_{i,j}^{n,m} \ , \\ \\ \mathcal{H} = \left(P^{-1} + \dfrac{3}{2}\lambda Id + \sigma_1\delta_1 A^E\mu_1 + \sigma_2\delta_2 B^E\mu_2 - \sigma_1\delta_1(\tilde{D}_1 + \dfrac{A_1^V}{\delta x})\delta_1 - \sigma_2\delta_2(\tilde{D}_2 + \dfrac{B_2^V}{\delta y})\delta_2 \right)^{n,m} \ , \end{array} \right.$$

where

$$\lambda = \frac{\Delta\tau}{\Delta t} \ , \quad \sigma_1 = \frac{\Delta\tau}{\delta x} \ , \quad \sigma_2 = \frac{\Delta\tau}{\delta y} \ .$$

In order to apply Von Neumann analysis, we have to linearize this system. This is done by "freezing" at a constant state the Jacobian matrices of the inviscid and viscous fluxes as well as the dissipation matrices, so that all these matrices do not depend on space nor time anymore. In such a case, the physical inviscid fluxes simply read :

$$\begin{aligned} (f^E)^{n,m} &= A^E \cdot w^{n,m} \ , \\ (g^E)^{n,m} &= B^E \cdot w^{n,m} \ . \end{aligned} \tag{3.2.1}$$

Considering higher-order extension, the numerical inviscid fluxes are linearized as follows :

$$\begin{aligned} \delta_1(F^E)_{i,j}^{n,m} &= \left[A^E \left(\delta_1\mu_1 - \frac{\varepsilon}{4}(1-\kappa)\delta_1^3\mu_1 \right) + \tilde{D}_1 \left((1-\varepsilon)\delta_1^2 - \frac{\varepsilon}{4}(1-\kappa)\delta_1^4 \right) \right] \cdot w_{i,j}^{n,m} \ , \\ \delta_2(G^E)_{i,j}^{n,m} &= \left[B^E \left(\delta_2\mu_2 - \frac{\varepsilon}{4}(1-\kappa)\delta_2^3\mu_2 \right) + \tilde{D}_2 \left((1-\varepsilon)\delta_2^2 - \frac{\varepsilon}{4}(1-\kappa)\delta_2^4 \right) \right] \cdot w_{i,j}^{n,m} \ , \end{aligned} \tag{3.2.2}$$

while the numerical viscous fluxes read :

$$\begin{aligned} \delta_1(F^V)_{i,j}^{n,m} &\approx \left[A_1^V \cdot \frac{\delta_1^2}{\delta x} + A_2^V \cdot \frac{\delta_1\delta_2\mu_1\mu_2}{\delta y} \right] \cdot w_{i,j}^{n,m} \ , \\ \delta_2(G^V)_{i,j}^{n,m} &\approx \left[B_1^V \cdot \frac{\delta_2\delta_1\mu_2\mu_1}{\delta x} + B_2^V \cdot \frac{\delta_2^2}{\delta y} \right] \cdot w_{i,j}^{n,m} \ . \end{aligned} \tag{3.2.3}$$

Using these expressions, the linearized system can be written under the following form :

$$\mathcal{H} \cdot \Delta w_{i,j}^{n,m} = -\mathcal{K} \cdot w_{i,j}^{n,m} + \frac{3\lambda}{2} w_{i,j}^{n} + \frac{\lambda}{2} \Delta w_{i,j}^{n-1} \quad . \tag{3.2.4}$$

with

$$\left\{\begin{array}{l} \mathcal{K} = \mathcal{K}^{III} = \frac{3}{2}\lambda Id + \dot{A}^E(\delta_1\mu_1 - \frac{\varepsilon}{4}(1-\kappa)\delta_1^3\mu_1) - ((1-\varepsilon)\dot{\tilde{D}}_1 + \dot{A}_1^V)\delta_1^2 - \dot{\tilde{D}}_1\frac{\varepsilon}{4}(1-\kappa)\delta_1^4 \\ \qquad\qquad +\dot{B}^E(\delta_2\mu_2 - \frac{\varepsilon}{4}(1-\kappa)\delta_2^3\mu_2) - ((1-\varepsilon)\dot{\tilde{D}}_2 + \dot{B}_2^V)\delta_2^2 - \dot{\tilde{D}}_2\frac{\varepsilon}{4}(1-\kappa)\delta_2^4 \\ \qquad\qquad -\dot{S}^V\delta_1\mu_1\delta_2\mu_2 \\ \\ \mathcal{H} = P^{-1} + \frac{3}{2}\lambda Id + \dot{A}^E\delta_1\mu_1 + \dot{B}^E\delta_2\mu_2 - (\dot{\tilde{D}}_1 + \dot{A}_1^V)\delta_1^2 - (\dot{\tilde{D}}_2 + \dot{B}_2^V)\delta_2^2 \end{array}\right. \tag{3.2.5}$$

where the following notations were used :

$$\begin{array}{l} \dot{A}^E = \sigma_1 A^E \ , \quad \dot{A}_1^V = \sigma_1\frac{A_1^V}{\delta x} \ , \quad \dot{A}_2^V = \sigma_1\frac{A_2^V}{\delta y} \ , \\ \dot{B}^E = \sigma_2 B^E \ , \quad \dot{B}_1^V = \sigma_2\frac{B_1^V}{\delta x} \ , \quad \dot{B}_2^V = \sigma_2\frac{B_2^V}{\delta y} \ , \\ \dot{S}^V = \dot{A}_2^V + \dot{B}_1^V \ . \end{array}$$

If the first-order upwind scheme is used, the linearized explicit operator $\mathcal{K}$ takes the simple form :

$$\mathcal{K} = \mathcal{K}^I = \frac{3}{2}\lambda Id + \dot{A}^E\delta_1\mu_1 + \dot{B}^E\delta_2\mu_2 - (\dot{\tilde{D}}_1 + \dot{A}_1^V)\delta_1^2 - (\dot{\tilde{D}}_2 + \dot{B}_2^V)\delta_2^2 - \dot{S}^V\delta_1\mu_1\delta_2\mu_2$$

In the remainder of this chapter, we will mainly focus on the third-order explicit stage $\mathcal{K}_{III}$ since it is the one used in practice; however, the implicit operator $\mathcal{H}$ will be the one deduced from the first-order upwind scheme (see 3.2.5), for cost reason as explained in chapter 1. A few words will be said about the impact of this choice, with reference to the case where the implicit stage is consistent with a first-order explicit stage defined by $\mathcal{K}_I II$.

3.3 Fourier Transform

In order to perform a Von Neumann analysis of the general linear scheme (3.2.4), it must be transformed into the Fourier space. The appendix B at the end of the thesis provides more details on the Fourier transform for the interested reader. The present section goes straight to the point of giving the Fourier transform of the general scheme (3.2.4)-(3.2.5) that will be used hereafter in the Von Neumann analysis. An angle of phase or reduced wavenumber is associated with each geometric direction :

$$\xi = \frac{2\pi}{\lambda_x}\cdot\delta x \ , \quad \eta = \frac{2\pi}{\lambda_y}\cdot\delta y \ , \tag{3.3.1}$$

where λ_x and λ_y denote the wavelengths in each direction. When applied to the space difference operators, the Fourier transformation leads to :

$$\widehat{\delta_1\mu_1} = \underline{\mathrm{i}}\ s_1 \ , \quad \widehat{\delta_2\mu_2} = \underline{\mathrm{i}}\ s_2 \ , \quad \widehat{\delta_1}^2 = -2z_1 \ , \quad \widehat{\delta_2}^2 = -2z_2 \ ,$$

where

$$s_1 = \sin(\xi) \ , \quad s_2 = \sin(\eta) \ , \quad z_1 = (1 - \cos(\xi)) \ , \quad z_2 = (1 - \cos(\eta)) \ .$$

Since constant functions are not represented in the Fourier space, the contribution of states w^n and w^{n-1}, which remain constant during the convergence of the dual-time stepping scheme, vanish. By transforming each side of equation (3.2.4) we eventually obtain :

$$H \cdot \Delta \widehat{w}_{i,j}^{n,m} = -K \cdot \widehat{w}_{i,j}^{n,m} \quad , \tag{3.3.2}$$

where H, K and $\widehat{w}^{n,m}$ denote the transformed functions of $\mathcal{H}$, $\mathcal{K}$ and $w^{n,m}$ respectively. In the present case of the dual-time implicit scheme, operators H and K read :

$$\left\{ \begin{array}{rcl} K & = & \frac{3}{2}\lambda Id \ + \ 2\left((1-\varepsilon)\dot{\tilde{D}}_1 \ + \ \dot{A}_1^V\right) z_1 \ + \ 2\left((1-\varepsilon)\dot{\tilde{D}}_2 \ + \ \dot{B}_2^V\right) z_2 \\ & & + \ \epsilon \ (1-\kappa)\dot{\tilde{D}}_1 \ z_1^2 \ + \ \epsilon \ (1-\kappa)\dot{\tilde{D}}_2 \ z_2^2 \ - \ \dot{S}^V s_1 s_2 \\ & & + \ \underline{\mathrm{i}} \ \left(\dot{A}^E \ s_1 \ \left(1 + \frac{\epsilon}{2} \ (1-\kappa) \ z_1\right) + \dot{B}^E \ s_2 \ \left(1 + \frac{\epsilon}{2} \ (1-\kappa) \ z_2\right)\right) \\ H & = & P^{-1} + \ \frac{3}{2}\lambda Id \ + \ 2\left(\dot{\tilde{D}}_1 \ + \ \dot{A}_1^V\right) \ z_1 \ + \ 2\left(\dot{\tilde{D}}_2 \ + \ \dot{B}_2^V\right) \ z_2 \ + \ \underline{\mathrm{i}}\left(\dot{A}^E s_1 + \dot{B}^E s_2\right) \end{array} \right. \tag{3.3.3}$$

If the first-order upwind scheme is used, the Fourier transform of the explicit operator K^I is given by :

$$K = \frac{3}{2}\lambda Id + 2 \ (\dot{\tilde{D}}_1 + \dot{A}_1^V) \ z_1 + 2 \ (\dot{\tilde{D}}_2 + \dot{B}_2^V) \ z_2 - \dot{S}^V s_1 s_2 + \underline{\mathrm{i}} \ (\dot{A}^E s_1 + \dot{B}^E s_2)$$

3.4 Amplification factor

In order to study the stability properties of the implicit upwind scheme, we have to define the amplification factor. For any pseudo-time level (n, m), the following relation between $\widehat{w}^{n,m+1}$ and $\widehat{w}^{n,m}$ can be written :

$$\widehat{w}^{n,m+1}(\xi, \eta) = G(\xi, \eta) \cdot \widehat{w}^{n,m}(\xi, \eta) \tag{3.4.1}$$

where G denotes the amplification matrix which depends on wavenumbers ξ, η as well as on the Jacobian matrices and dissipation matrices. The linear stability of the scheme is known from the amplification factor which is the spectral radius $\rho(G)$ of this amplification matrix G : namely, if $\rho(G) < 1$ the errors in the system are damped whereas if $\rho(G) > 1$ the errors are amplified leading to instability. More importantly, the magnitude of the amplification factor allows to estimate the convergence rate of the numerical scheme. Indeed, if $\rho(G) \to 0$, a fast damping of all the error modes can be expected and consequently a fast convergence, while if $\rho(G) \to 1$, a poor damping and hence a poor convergence rate is bound to be obtained. In practice, to analyze the performances of the scheme, a finite number of values for the wave numbers is fixed (in other words the wave numbers plane is discretized with a given level of refinement) then

the amplification factor is computed for each combination of these values. This computation is not performed analytically when dealing with a linearized system of equation but using our own Fourier analysis code. In this chapter, our attention will be restricted to the case of the linearized (preconditioned) Euler equations because our studies have shown that the conclusions drawn in that case regarding the most efficient implicit treatment could be extended with confidence to the case of viscous and unsteady flows.

3.5 Analysis of the Direct scheme

The first method to be analyzed is the one for which the system associated with the implicit stage is solved directly, *i.e.* without approximation. This so-called direct solver does not introduce any error (contrarily to approximate factorization method for instance), so it is expected to provide the best amplification factor. Hence, it will stand as our point of reference, as far as convergence rate or intrinsic efficiency is concerned, in the remainder of the chapter. The amplification matrix associated to this scheme reads :

$$G_* = Id - H^{-1} \cdot K \tag{3.5.1}$$

For a given implicit and explicit stage, the amplification factor of the direct solver (or any other implicit treatment besides) applied to the linearized Euler equations depends on a limited number of parameters, through the definition of H and K : the wave-numbers of course, the ratio σ_1, σ_2, the Jacobian matrices A^E and B^E and the dissipation matrices corresponding to the Roe upwind scheme, that is given by (1.2.8) for compressible flows and by (1.3.30) in the low-Mach number preconditioned case. For the linearized Euler equations, A^E, B^E, $\tilde{D}_1$ and $\tilde{D}_2$ are constant matrices that essentially depend on the Mach number M and the flow direction v/u (the speed of sound c and the density ρ can be kept constant at a reference value without loss of generality while the ratio of specific heats γ is such that $\gamma = 1.4$; for the non-dimensional Euler equations the choice $\rho = 1$, $c = 1$ is retained). Moreover, through the ratio σ_1 and σ_2, H and K also depend on the aspect ratio $AR = \delta y/\delta x$ and on the CFL number introduced in the definition of the time-step :

$$\Delta t = CFL \cdot \min(\frac{\delta x}{\lambda_1^+}, \frac{\delta y}{\lambda_2^+}) \tag{3.5.2}$$

where λ_1^+ and λ_2^+ denote the spectral radii of A^E and B^E ($\lambda_1^+ = |u| + c$ and $\lambda_2^+ = |v| + c$). When preconditioning is applied, the time step is defined using the spectral radius of the preconditioned Jacobians PA^E and PB^E. Using this time step definition, the parameters σ_1 and σ_2 can be computed :

$$\begin{cases} \dot{A}^E = \dfrac{\Delta t}{\delta x} A^E = CFL \cdot \min(\dfrac{1}{\lambda_1^+}, \dfrac{AR}{\lambda_2^+}) \cdot A^E(v/u, M) \\ \dot{B}^E = \dfrac{\Delta t}{\delta y} B^E = CFL \cdot \min(\dfrac{1/AR}{\lambda_1^+}, \dfrac{1}{\lambda_2^+}) \cdot B^E(v/u, M) \end{cases} \tag{3.5.3}$$

Since $\rho(A^E)$, $\rho(B^E)$ (or their preconditioned equivalents) depend themselves on v/u and M, the quantities $\dot{A}^E$ and $\dot{B}^E$ depend eventually on the Mach number M, flow direction v/u, CFL

number and aspect ratio $\delta y/\delta x$; the dissipation matrices are also functions of these same parameters. In the preconditioned case, P, $\tilde{\dot{D}}_1 = \sigma_1 \tilde{D}_1$ and $\tilde{\dot{D}}_2 = \sigma_2 \tilde{D}_2$ can also be expressed in terms of these parameters since the preconditioning parameter β, expressed using the global definition (1.3.2) depends exclusively on the (fixed) Mach number. Summing things up, it is thus possible to write :

$$G_*(\xi,\eta;M,CFL,v/u,AR) = Id - H^{-1}(\xi,\eta;M,CFL,v/u,AR)\cdot K(\xi,\eta;M,CFL,v/u,AR)$$

Preliminary studies have shown that the most influential parameters in the case of the linearized Euler equations are the Mach number for the physics and the CFL number for the numerics. In the remainder of this chapter, it will therefore be assumed that the aspect ratio is equal to 1 (which is a reasonable assumption for the grids used in inviscid flow computations) and the flow direction is diagonal with respect to the Cartesian grid ($v/u = 1$ which makes the flow truly $2D$). Several types of analysis can be performed on the amplification factor :

- local stability and efficiency analysis :
  for a given set of physical (M, v/u) and numerical (CFL, AR) parameters, the spectral radius $\rho(G_*)(\xi,\eta)$ is computed with a sweep on the wave-numbers $\xi \in [0,\pi]$, $\eta \in [0,\pi]$; if there exists at least a value of (ξ,η) for which $\rho(G_*) > 1$ the scheme is unstable for the fixed values of the physical and numerical parameters considered. Apart from the couple $(\xi,\eta) = (0,0)$ for which $\rho(G_*) = 1$ for consistency reason, achieving the smallest possible value of $\rho(G_*)$ for all $(\xi,\eta) \neq (0,0)$ ensures a fast damping (or a high intrinsic efficiency) of the scheme under study.

- global stability analysis :
  the quantity $\rho_*^{max}(M,CFL) = \max_{\xi\in[0,\pi],\eta\in[0,\pi]}(\rho(G_*(\xi,\eta;M,CFL)))$ is computed with a sweep on the physical parameter M and the numerical parameter CFL. If the scheme is stable for a given (M,CFL) couple, this maximal value is equal to 1 (obtained for $\xi = \eta = 0$). So that, if a scheme is globally stable for the whole range of Mach and CFL numbers considered in the study, $\rho_*^{max}(M,CFL) = 1\,\forall M,\,\forall CFL$. If the scheme is unstable for some (M,CFL) couples, values larger than 1 for $\rho_*^{max}(M,CFL)$ appear in the study.

- global efficiency analysis :
  the quantity $\bar{\rho}_*(M,CFL)$ is the mean value of $\rho(G_*(\xi,\eta;M,CFL))$ computed when ξ and η vary between 0 and π. It is computed with a sweep on the physical parameter M and the numerical parameter CFL. This mean value may of course be lower than 1 even though the scheme is unstable and exhibits $\rho(G(\xi,\eta)) > 1$ for some values of (ξ,η); so the quantity $\bar{\rho}_*(M,CFL)$ does not provide any information on the scheme stability. However, it completes very usefully the previous global stability analysis based on $\rho_*^{max}(M,CFL)$. Indeed, if the scheme is stable for a given (M,CFL) couple ($\rho_*^{max}(M,CFL) \leq 1$), the mean value provides a quantitative information on the damping or intrinsic efficiency of the scheme : the slower the value of $\bar{\rho}_*(M,CFL)$, the better the damping offered by the numerical treatment for this couple of Mach and CFL numbers.

Naturally, the local or global analysis carried out for the direct solver can be also performed, with the same interpretation for the approximate solution methods considered in the present work. Let us start with a study of the direct solver amplification factor $\rho(G_*)$ when M and CFL are free parameters (while $v/u = 1$ and $AR = 1$).

3.5.1 Effects of inconsistency between implicit and explicit stages

The effect on efficiency of coupling a simple first-order implicit operator $\mathcal{H}^I = I + \mathcal{K}^I$ with a third-order explicit operator $\mathcal{K}^{III}$ has already been analyzed in the literature (for instance by Pulliam *et al.* in [64]). Since this simple non-consistent choice is retained throughout this work, it seems interesting to summarize the consequences of such a choice for the whole range of Mach number. First of all, both III/III and I/III schemes are found to be unconditionally stable when exactly solved, be it in their non-preconditioned or low-Mach preconditioned version. The efficiency of both schemes in the non-preconditioned case is analyzed through the plots of $\bar{\rho}_*$ in function of M and CFL displayed in figure 3.5.1 :

- when large CFL numbers are used, the consistent choice III/III provides an excellent damping as long as the Mach number does not become too low. When $M \to 0$, the error damping goes to 1 (almost no damping at all hence a very low convergence rate) whatever the value of the CFL number. This well-known behavior is typical of the low-Mach number issue with upwind compressible flow solvers.
- the non-consistent choice I/III displays the same weakness for low values of the Mach number. Moreover, for large values of the CFL number, the mean value of the amplification factor does not tend to zero any longer but to a finite value (typically above 0.5 for the whole range of Mach number) : this loss of efficiency is a direct consequence of the fact $\mathcal{H}^I = I + \mathcal{K}^I \neq I + \mathcal{K}^{III}$.

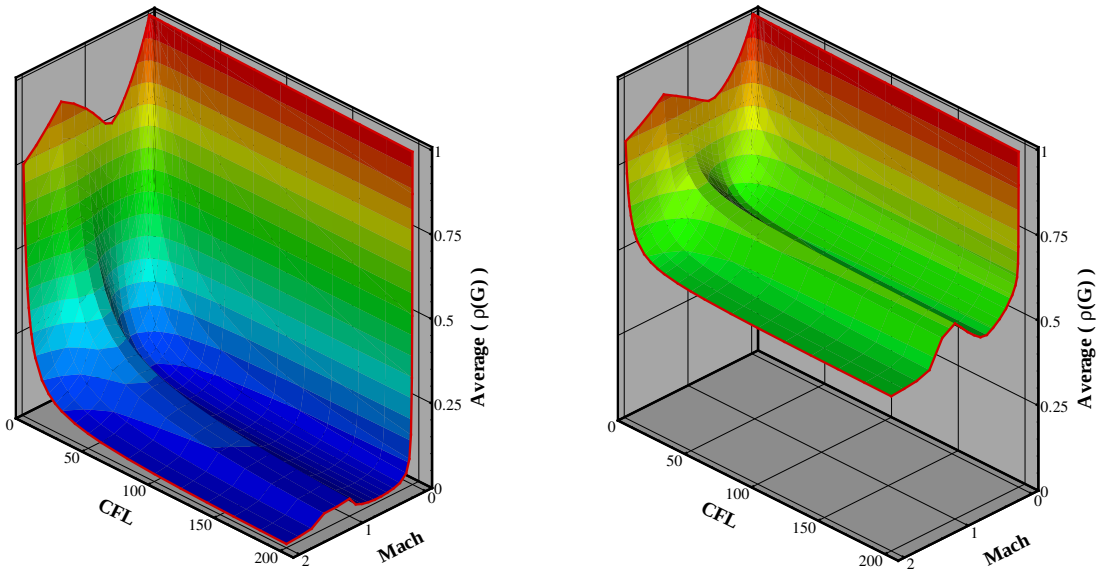


Figure 3.5.1: Average amplification factor for the Direct scheme. Left : consistent scheme III/III. Right : Inconsistent scheme I/III.

Considering the behavior of the average value of the amplification factor, it can seem unnecessary to use high CFL numbers for the inconsistent scheme I/III. In order to investigate this point, a local study, that is a study of the amplification factor versus the wavenumbers ξ and η for a specific Mach number and for two different CFL numbers, is performed. A subsonic case ($M = 0.5$) is selected (to avoid any interference with low-Mach effects that will be dealt with below)

and a comparison is made between the amplification factor of the Direct Scheme obtained with $CFL = 100$ and $CFL = 10^6$. The results are presented in figure 3.5.2. As expected, the global shape of the amplification factor is quite the same for $CFL = 100$ and $CFL = 10^6$. Nevertheless, in the low-frequency region, that is for wave-numbers close to zero, there is a non-negligible gain of efficiency achieved by using a very high CFL. In appendix B, we explain why the efficiency of a method in the low-frequency zone is in fact essential for the convergence performance. Thus, in spite of an average value $\bar{\rho}_*$ almost independent of CFL once this parameter is large enough, there is still a benefit to run at infinite CFL numbers.

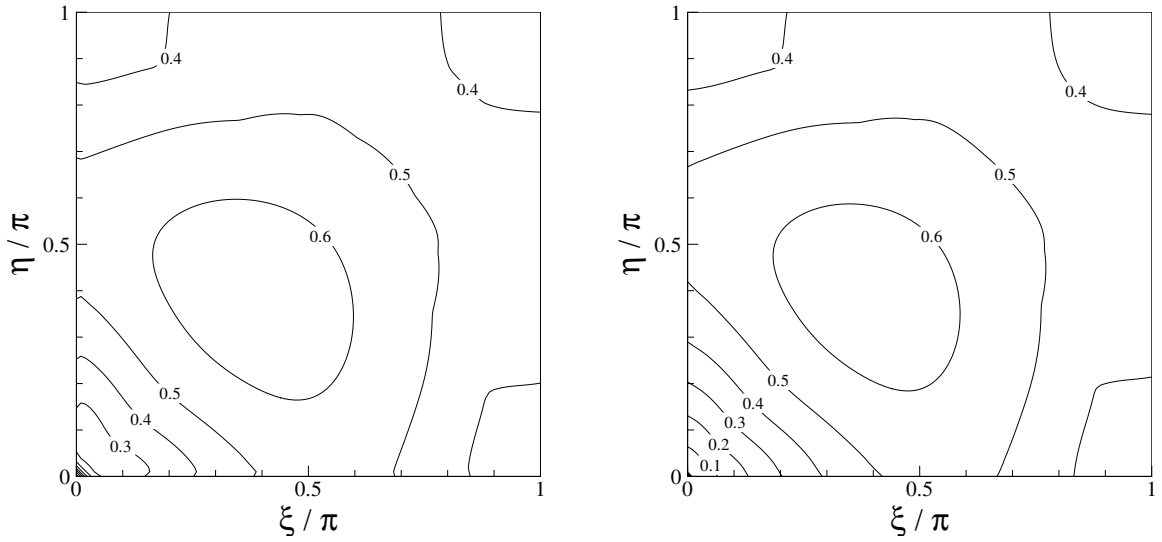


Figure 3.5.2: Amplification factor for the Direct Scheme at $M = 0.5$. Left : $CFL = 100$. Right : $CFL = 10^6$.

3.5.2 Low Mach number effects

It has been noted on figure 3.5.1 that the average amplification factor of the Direct Scheme goes to unity when the Mach number goes to zero, leading to a poor damping of the errors. This problem motivated to use of low-Mach preconditioning methods in order to improve the efficiency of the compressible solver in this regime. The Direct I/III scheme using Turkel preconditioner recovers a very good efficiency for small Mach numbers (*cf.* figure 3.5.3) and remains stable for all CFL numbers. Note the results obtained for the Direct III/III scheme demonstrate that low-Mach preconditioning is essential to provide truly excellent damping and fast convergence for the whole range of Mach number when large CFL numbers are used (but, alas, the consistent III/III scheme is clearly too expensive both in terms of unit computational cost and memory storage). In the manner of the previous section, a local analysis can be carried out in order to analyze more closely the gain of efficiency obtained with the preconditioned Direct I/III Scheme with respect to the non-preconditioned version. A low-Mach case ($M = 10^{-3}$) is now selected and a high CFL number ($CFL = 10^6$) is retained since it has been previously shown that such a choice enables to increase the damping over the low-frequency part of the spectrum. The results are presented in figure 3.5.4 :

- for the non-preconditioned scheme, there is still a good damping in the high-wavenumber

regions but the amplification factor in the low wavenumber area becomes very close to unity, leading therefore to poor damping and convergence (low intrinsic efficiency).

- on the other hand, the preconditioned Direct Scheme provides a very good damping over the entire wavenumber domain. Indeed, the results are very close to the original scheme under standard "compressible flow" conditions given in figure 3.5.2. Similar analyzes are made by Merkle and Venkateswaran in [81] and the reader can refer to this work for instance for further studies about preconditioning techniques (viscous flow, high-aspect ratio or unsteady flow for instance).

Since the Direct Solver efficiency properties represent an ideal to achieve at the lowest possible cost using an approximate iterative solution technique, only the preconditioned implicit stage will be considered in the remainder of this chapter.

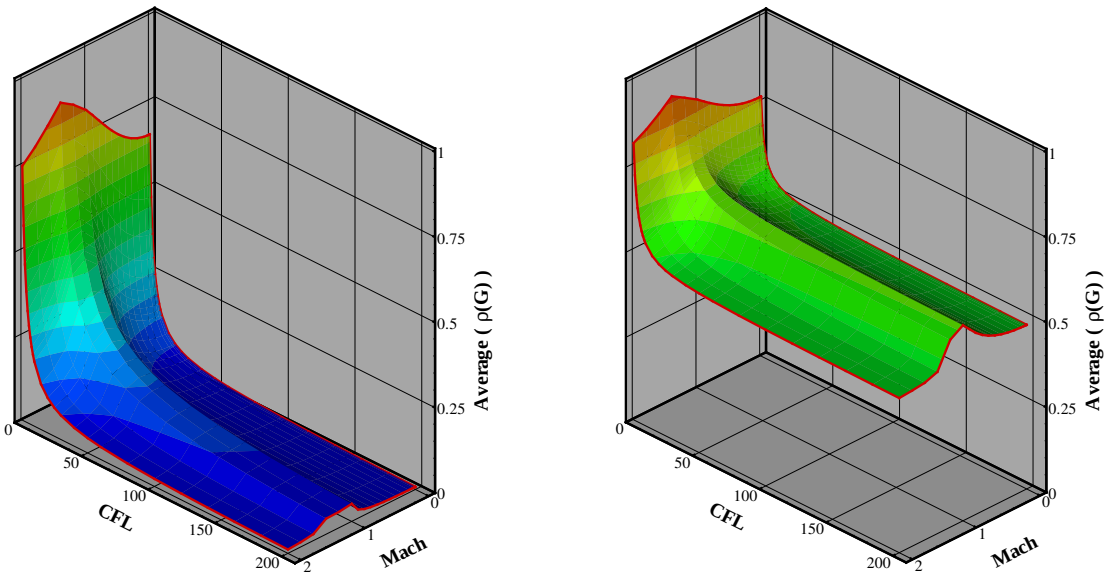


Figure 3.5.3: Mean value of the amplification factor for the Direct scheme using the Turkel preconditioner. Left :consistent scheme III/III. Right : Inconsistent scheme I/III.

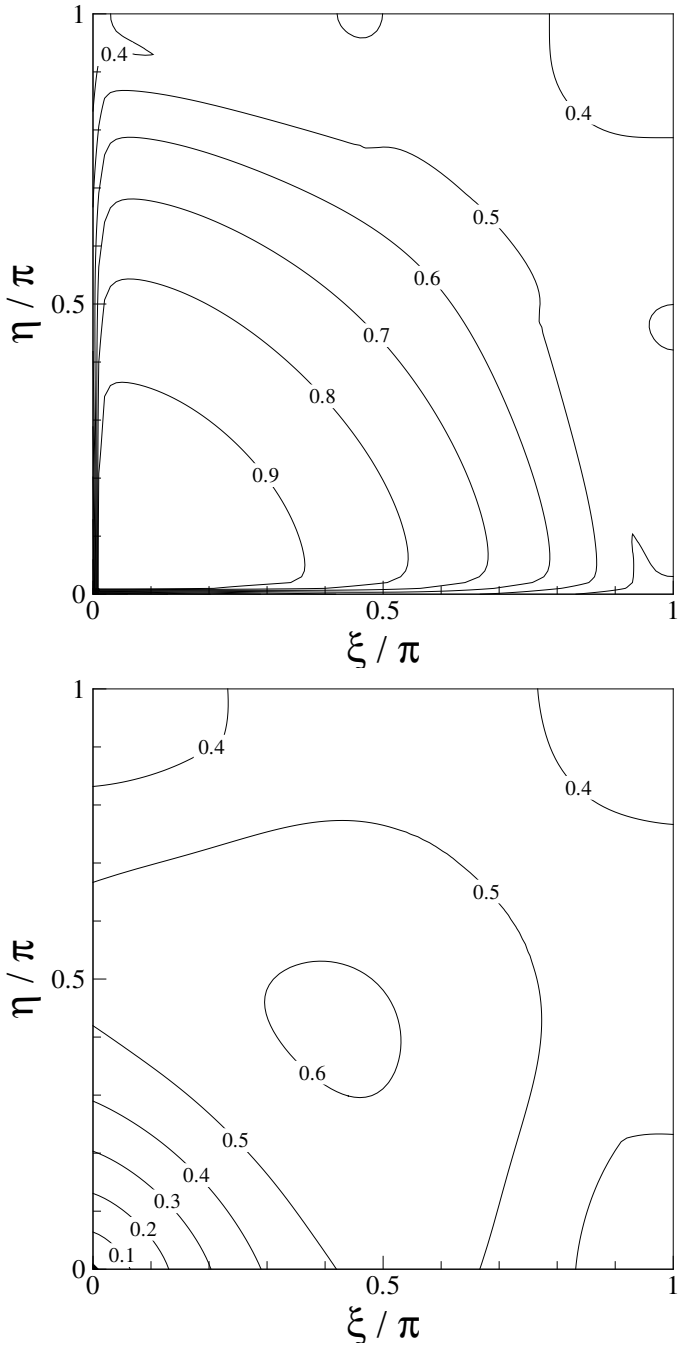


Figure 3.5.4: Amplification factor for the Direct Scheme at $M = 10^{-3}$ with $CFL = 10^6$. Top : No Preconditioning. Bottom : Turkel Preconditioning.

3.6 Analysis of approximate iterative implicit treatment

In chapter 2, several approximate inversion methods have been described to solve the linear system (1.2.18). The presentation of these methods has been precisely based on a linear system viewpoint, introducing matrices M_1', M_2' and D to split the large-banded matrix M associated with (1.2.18) as well as lower and upper matrices L_1, L_2, U_1 and U_2. The present section is devoted to the presentation of a unifying viewpoint on all these methods that will make easier the Fourier analysis of these various implicit treatments. This unifying formulation is based on a difference operator viewpoint rather than a linear system viewpoint (though both are equivalent). Let us thus first consider the implicit scheme for low-Mach number unsteady flows (1.4.9) and let us focus on the implicit operator $\mathcal{H}$, which can be split into :

$$\mathcal{H} = D_0 + \mathcal{T}_1 + \mathcal{T}_2 \tag{3.6.1}$$

where D_0, $\mathcal{T}_1$ and $\mathcal{T}_2$ are difference operators given by :

$$\left\{ \begin{array}{l} D_0 = P^{-1} + \dfrac{3}{2}\lambda Id \ , \\ \\ \mathcal{T}_1 = \sigma_1 \left(\delta_1 A^E \mu_1 - \delta_1 (\tilde{D}_1 + \dfrac{A_1^V}{\delta x}) \delta_1 \right) \ , \\ \\ \mathcal{T}_2 = \sigma_2 \left(\delta_2 B^E \mu_2 - \delta_2 (\tilde{D}_2 + \dfrac{B_2^V}{\delta y}) \delta_2 \right) \ . \end{array} \right.$$

Assuming constant coefficients, that is considering a linear problem, the operators $\mathcal{T}_p$ become :

$$\left\{ \begin{array}{l} \mathcal{T}_1 = \dot{A}^E \delta_1 \mu_1 - (\dot{\tilde{D}}_1 + \dot{A_1^V}) \delta_1^2, \\ \\ \mathcal{T}_2 = \dot{B}^E \delta_2 \mu_2 - (\dot{\tilde{D}}_2 + \dot{B_2^V}) \delta_2^2 \ . \end{array} \right.$$

Alternately, the implicit operator $\mathcal{H}$ can be also split as follows :

$$\mathcal{H} = D + \mathcal{S}_1 + \mathcal{S}_2 \tag{3.6.2}$$

where D, $\mathcal{S}_1$ and $\mathcal{S}_2$ are defined as :

$$\left\{ \begin{array}{l} D = C^0 \ , \\ \\ \mathcal{S}_1 \cdot \Delta w_{i,j}^{n,m} = C_1^+ \cdot \Delta w_{i+1,j}^{n,m} + C_1^- \cdot \Delta w_{i-1,j}^{n,m} \ , \\ \\ \mathcal{S}_2 \cdot \Delta w_{i,j}^{n,m} = C_2^+ \cdot \Delta w_{i,j+1}^{n,m} + C_2^- \cdot \Delta w_{i,j-1}^{n,m} \ , \end{array} \right.$$

with

$$\left\{ \begin{array}{l} C_1^+ = (\dfrac{1}{2}\dot{A}^E - (\dot{\tilde{D}}_1 + \dot{A_1^V}))_{i+\frac{1}{2},j} = \dot{A}^-_{i+\frac{1}{2},j} \ , \\ \\ C_1^- = -(\dfrac{1}{2}\dot{A}^E + (\dot{\tilde{D}}_1 + \dot{A_1^V}))_{i-\frac{1}{2},j} = -\dot{A}^+_{i-\frac{1}{2},j} \ , \\ \\ C_2^+ = (\dfrac{1}{2}\dot{B}^E - (\dot{\tilde{D}}_2 + \dot{B_2^V}))_{i,j+\frac{1}{2}} = \dot{B}^-_{i,j+\frac{1}{2}} \ , \\ \\ C_2^- = -(\dfrac{1}{2}\dot{B}^E + (\dot{\tilde{D}}_2 + \dot{B_2^V}))_{i,j-\frac{1}{2}} = -\dot{B}^+_{i,j-\frac{1}{2}} \ , \\ \\ C^0 = P^{-1} + \dfrac{3}{2}\lambda Id + (\dot{A}^+_{i+\frac{1}{2},j} - \dot{A}^-_{i-\frac{1}{2},j}) + (\dot{B}^+_{i,j+\frac{1}{2}} - \dot{B}^-_{i,j-\frac{1}{2}}) \ . \end{array} \right.$$

For a linear problem, the operators $\mathcal{S}$ and the coefficient D can be simplified into :

$$\begin{cases} D = P^{-1} + \dfrac{3}{2}\lambda Id + 2(\dot{\tilde{D}}_1 + \dot{A}_1^V) + 2(\dot{\tilde{D}}_2 + \dot{B}_2^V) \ , \\ \mathcal{S}_1 = \dot{A}^- \mathcal{E}_1^+ - \dot{A}^+ \mathcal{E}_1^- \ , \\ \mathcal{S}_2 = \dot{B}^- \mathcal{E}_2^+ - \dot{B}^+ \mathcal{E}_2^- \ , \end{cases}$$

where the operators $\mathcal{E}_p$ are shift operators over one grid cell such that :

$$\begin{cases} \mathcal{E}_1^{\pm} \cdot \Delta w_{i,j} = \Delta w_{i\pm1,j} \ , \\ \mathcal{E}_2^{\pm} \cdot \Delta w_{i,j} = \Delta w_{i,j\pm1} \ . \end{cases}$$

The two splittings (3.6.1) and (3.6.2) will be useful to characterize approximate factorization and relaxation treatments once all these treatments put under the form of a general iterative scheme, as explained below. The Fourier transform H of the implicit operator $\mathcal{H}$ has been defined in (3.3.3); alternatively, if T_1, T_2 denote the Fourier transforms of $\mathcal{T}_1$ and $\mathcal{T}_2$, and similarly, S_1 and S_2 are the Fourier transforms of $\mathcal{S}_1$ and $\mathcal{S}_2$, the symbol H can also be expressed as :

$$H = D_0 + T_1 + T_2 = D + S_1 + S_2 \tag{3.6.3}$$

Making use in particular of the Fourier transformation of the difference operators $\mathcal{E}^{\pm}$:

$$\begin{cases} E_1^{\pm} = e^{\pm \underline{\mathrm{i}}\ \xi} \ , \\ E_2^{\pm} = e^{\pm \underline{\mathrm{i}}\ \eta} \ . \end{cases}$$

leads to the following definition of the symbols T_p and S_p :

$$\begin{cases} T_1 = 2\ (\dot{\tilde{D}}_1 + \dot{A}_1^V))z_1 + \underline{\mathrm{i}}\ \dot{A}^E s_1 \ , \\ T_2 = 2\ (\dot{\tilde{D}}_2 + \dot{B}_2^V))z_2 + \underline{\mathrm{i}}\ \dot{B}^E s_2 \ , \\ S_1 = -(\dot{\tilde{D}}_1 + \dot{A}_1^V))\cos(\xi) + \underline{\mathrm{i}}\ \dot{A}^E s_1 \ , \\ S_2 = -(\dot{\tilde{D}}_2 + \dot{B}_2^V))\cos(\eta) + \underline{\mathrm{i}}\ \dot{B}^E s_2 \ . \end{cases}$$

with

$$z_1 = (1 - \cos(\xi)) \ , \quad z_2 = (1 - \cos(\eta)) \ , \quad s_1 = \sin(\xi) \ , \quad s_2 = \sin(\eta) \ .$$

Keeping in mind these various definitions for future use, we are now going to introduce a general form for the iterative schemes under study. A general iterative scheme whose form covers both iterative relaxation methods (such as $ALJ(p)$, $ALGS(p)$, $PJ(p)$ or $SGS(p)$) and iterative factorization methods (such as $AF(p)$, $MAF(p)$) reads :

$$\mathcal{H}_\alpha \cdot \Delta w_{i,j}^{(l+1)} = -\Delta\tau \mathcal{R}_{i,j}^{n,m} - (\mathcal{H} - \mathcal{H}_\alpha) \cdot \Delta w_{i,j}^{(l)} \tag{3.6.4}$$

where $\Delta w^{(l+1)} = w^{(l+1)} - w^{n,m}$ and $\mathcal{H}_\alpha$ is a difference operator that depends on the choice of the iterative method. In fact, as pointed out in chapter 2, the operator $\mathcal{H}_\alpha$ is not necessarily

the same at each stage of the sub-iteration process, so that a more general form of the iterative scheme is :

$$\begin{aligned} &\alpha = 1,\ N \\ &\mathcal{H}_\alpha \cdot \Delta w_{i,j}^{(lN+\alpha)} = -\Delta\tau \cdot \mathcal{R}_{i,j}^{n,m} - (\mathcal{H} - \mathcal{H}_\alpha) \cdot \Delta w_{i,j}^{(lN+\alpha-1)} \end{aligned} \tag{3.6.5}$$

The full algorithm is then given by :

$$\left\{ \begin{array}{l} \Delta w_{i,j}^{(0)} = 0 \\ l = 0,\ p-1 \\ \qquad \alpha = 1,\ N \\ \qquad \mathcal{H}_\alpha \cdot \Delta w_{i,j}^{(lN+\alpha)} = -\Delta\tau \cdot \mathcal{R}_{i,j}^{n,m} - (\mathcal{H} - \mathcal{H}_\alpha) \cdot \Delta w_{i,j}^{(lN+\alpha-1)} \\ \Delta w_{i,j}^{n,m} = \Delta w_{i,j}^{(pN)} \end{array} \right. \tag{3.6.6}$$

The expression of operators $\mathcal{H}_\alpha$ for the different iterative methods previously introduced will be detailed in the next paragraphs. First, the Fourier transform is applied to this general iterative algorithm in order to obtain a closed-form expression for the associated amplification factor :

$$\left\{ \begin{array}{l} \Delta\tau \cdot \widehat{\mathcal{R}}^{n,m} = K \cdot \widehat{w}^{n,m} \\ \Delta\widehat{w}^{(0)} = 0 \\ l = 0,\ p-1 \\ \qquad \alpha = 1,\ N \\ \qquad H_\alpha \cdot \Delta\widehat{w}^{(lN+\alpha)} = -K \cdot \widehat{w}^{n,m} - (H - H_\alpha) \cdot \Delta\widehat{w}^{(lN+\alpha-1)} \\ \Delta\widehat{w}^{n,m} = \Delta\widehat{w}^{(pN)} \end{array} \right. \tag{3.6.7}$$

At each step of the sub-iterative process, a provisional amplification matrix can be defined :

$$\forall i \in [|0\ ,\ pN|] \quad \widehat{w}^{(i)} = G_i \cdot \widehat{w}^{n,m}$$

Obviously, the choice $\Delta w^{(0)} = 0$ yields $G_0 = Id$ and the amplification matrix resulting from the global algorithm is :

$$G = G_{pN}$$

We want to express this matrix as a function of the Fourier symbols H, K and H_α. We will now recall the analysis presented in [39], [19]. Let us write the iterative scheme using the provisional matrix G_i :

$$H_\alpha\ (G_{lN+\alpha} - Id) = -K - (H - H_\alpha)\ (G_{lN+\alpha-1} - Id)$$

Now let us introduce in this expression the amplification matrix of the Direct Scheme $G_* = Id - H^{-1} \cdot K$; immediate calculations yield :

$$G_{lN+\alpha} = G_{lN+\alpha-1} - H_\alpha^{-1}\ H\ G_{lN+\alpha-1} + H_\alpha^{-1}\ H\ G_*$$

Subtracting G_* from the LHS and RHS leads to :

$$(G_{lN+\alpha} - G_*) = (Id - H_\alpha^{-1}\ H)\ (G_{lN+\alpha-1} - G_*)$$

When α varies from 1 to N, we obtain :

$$(G_{(l+1)N} - G_*) = \prod_{\alpha=1}^{N} (Id - H_\alpha^{-1}\ H)\ (G_{lN} - G_*)$$

Eventually, when the sub-iterative process is completed after p iterations, we have :

$$(G_{pN} - G_*) = \left[\prod_{\alpha=1}^{N} (Id - H_\alpha^{-1}\ H)\right]^p\ (Id - G_*)$$

The amplification matrix of the general iterative scheme is therefore defined by :

$$\begin{cases} G = G_* + V^p\ (Id - G_*) \\ \\ V = \prod_{\alpha=1}^{N} (Id - H_\alpha^{-1}\ H) \end{cases} \tag{3.6.8}$$

This formula enables us to quantify the impact on intrinsic efficiency of the two levels which drive the efficiency of an iterative method : firstly, the exact implicit scheme should provide the best damping, that is the best amplification matrix G_*; secondly, the iterative method should converge as fast as possible to this exact solver. According to formula (3.6.8), for a given exact implicit stage (such as the block one studied in the previous section), the convergence rate of an approximate solver depends on the coefficient V which is itself a function of the exact implicit stage Fourier transform H and of the Fourier transform(s) H_α specific to the chosen approximate treatment. This coefficient V should be as small as possible so as to provide the best damping for a reduced number of sub-iterations. If $H_\alpha = H$ then $V = 0$ and the amplification factor of the Direct Scheme which provides the best damping is naturally recovered. In the following sections, formula (3.6.8) will be used to analyze the amplification factor of the iterative implicit treatments described in chapter 2.

3.7 Alternate Direction Implicit Method

The ADI or AF method improved by a sub-iterative process aimed at canceling the factorization error can be clearly cast into the general form (3.6.6) using the single implicit operator $\mathcal{H}_\alpha$ ($\alpha = 1$) :

$$\mathcal{H}_{ADI} = (D_0 + \mathcal{T}_1) \cdot D_0^{-1} \cdot (D_0 + \mathcal{T}_2) \tag{3.7.1}$$

that is

$$\mathcal{H}_{ADI} = \mathcal{H} + \mathcal{T}_1 \cdot D_0^{-1} \cdot \mathcal{T}_2 \tag{3.7.2}$$

The amplification matrix associated with this method is immediately deduced from (3.6.6) :

$$\begin{cases} G_{ADI} = G_* + V_{ADI}^p\ (Id - G_*) \\ \\ V_{ADI} = (Id - H_{ADI}^{-1}\ H) \end{cases} \tag{3.7.3}$$

with

$$H_{ADI} = (D_0 + T_1) \cdot D_0^{-1} \cdot (D_0 + T_2) = H + T_1 \cdot D_0^{-1} \cdot T_2 \tag{3.7.4}$$

One can notice that when a single sub-iteration is performed ($p = 1$), the original ADI scheme is recovered with the corresponding amplification factor $G_{ADI} = Id - H_{ADI}^{-1} \cdot K$. Let us now study the properties of the $ADI(p)$ or $AF(p)$ method using Von Neumann analysis.

3.7.1 Global analysis

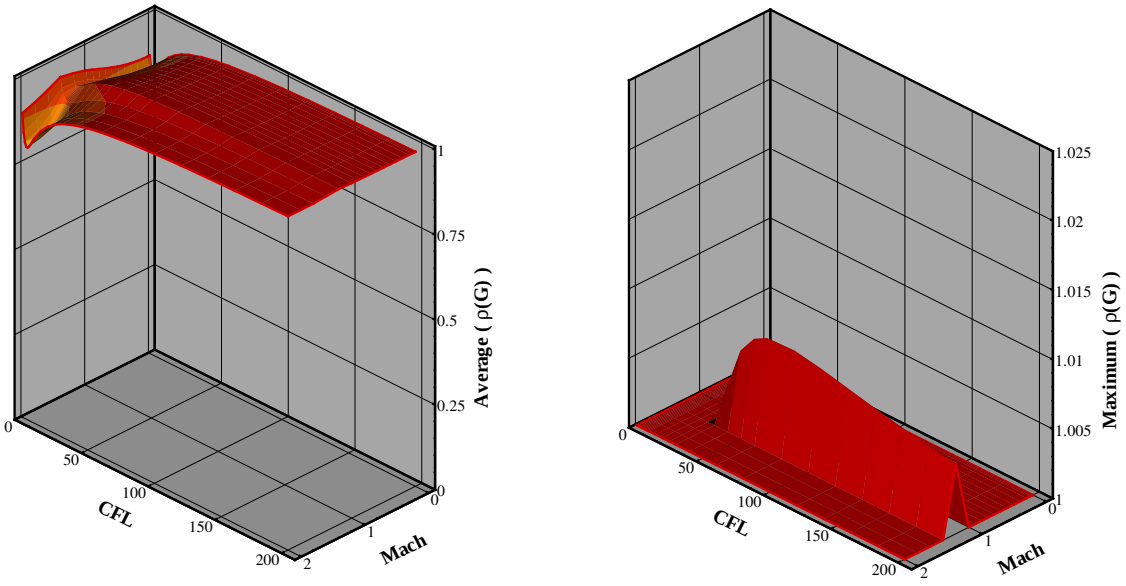


Figure 3.7.1: Average (left) and maximum (right) amplification factor of the non-iterative ADI scheme.

The average $\bar{\rho}_{ADI}$ and maximum ρ_{ADI}^{max} amplification factor for the non-iterative ADI scheme are plotted in figure (3.7.1) as a function of the Mach number and the CFL number. The very well-known behavior of this method, namely a very poor damping for high CFL numbers due to a dominant factorization error, is made clear by these plots. We observe that the optimal CFL value is around 5 which is also well-established by several studies [64]. The sub-iterative process that was previously described ($ADI(p)$ or $AF(p)$) improves the amplification factor. The ADI scheme with 20 sub-iterations provides a better damping of the error (*cf.* figure 3.7.2). However, this damping still deteriorates for high CFL number and we even note instabilities for $CFL > 100$. There is still an optimum CFL number whose value is about 20. We showed previously that for such value of the CFL number, the Direct Scheme does not provide the best damping, so the efficiency of the ADI method, even with a sub-iterative process, is not satisfactory since it does not allow anyhow to recover the convergence rate offered by the exact solver.

3.7.2 Local analysis

Studying more precisely the properties of the ADI scheme requires a local analysis. We choose a low-Mach case ($M = 10^{-3}$) and we use the ADI scheme with 20 sub-iterations. Since the

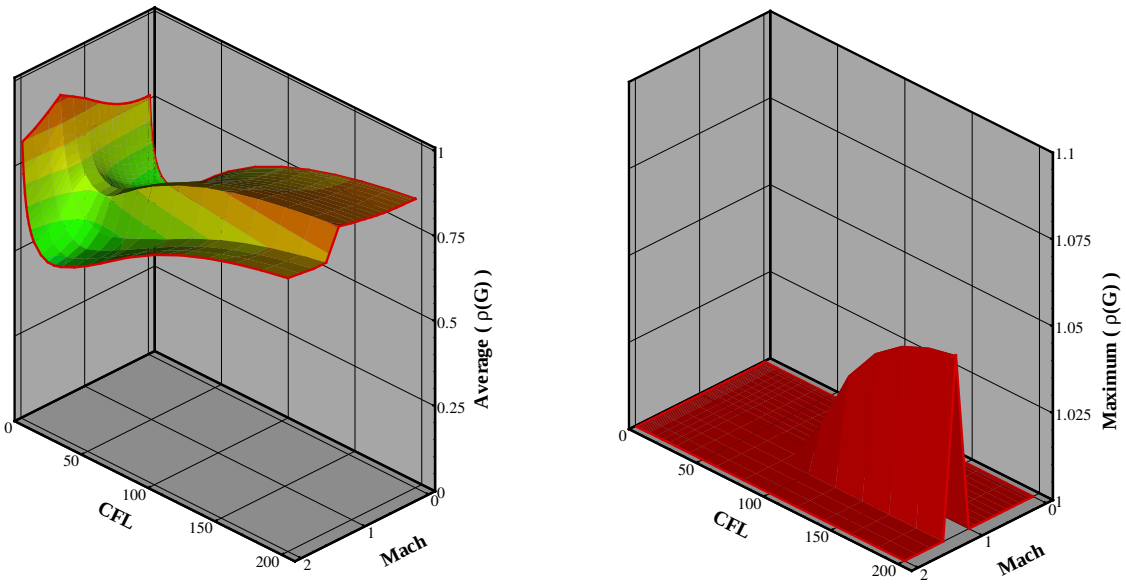


Figure 3.7.2: Average (left) and maximum (right) amplification factor of the non-iterative ADI scheme.

global analysis showed that the optimum CFL number was 20 in that case, we compare this method with the Direct Scheme at CFL 20. The results are presented in figure 3.7.3. The amplification factor of the ADI scheme is close to the Direct Scheme for the low wavenumbers. Nevertheless, we note that 20 sub-iterations are not sufficient to obtain the same efficiency than the Direct Scheme, especially for the high frequency regions. Further analyzes have shown that 50 sub-iterations have to be made, at least, to recover this efficiency over the entire wavenumber domain. This result confirms that ADI scheme, even with a sub-iteration process, is inefficient. First, it requires a finite time-step for which the error damping will be anyway sub-optimal (with respect to the one that can be expected from the exact treatment); second, it needs a large amount of sub-iterations to be as efficient as the Direct Scheme, even for this sub-optimal time-step value.

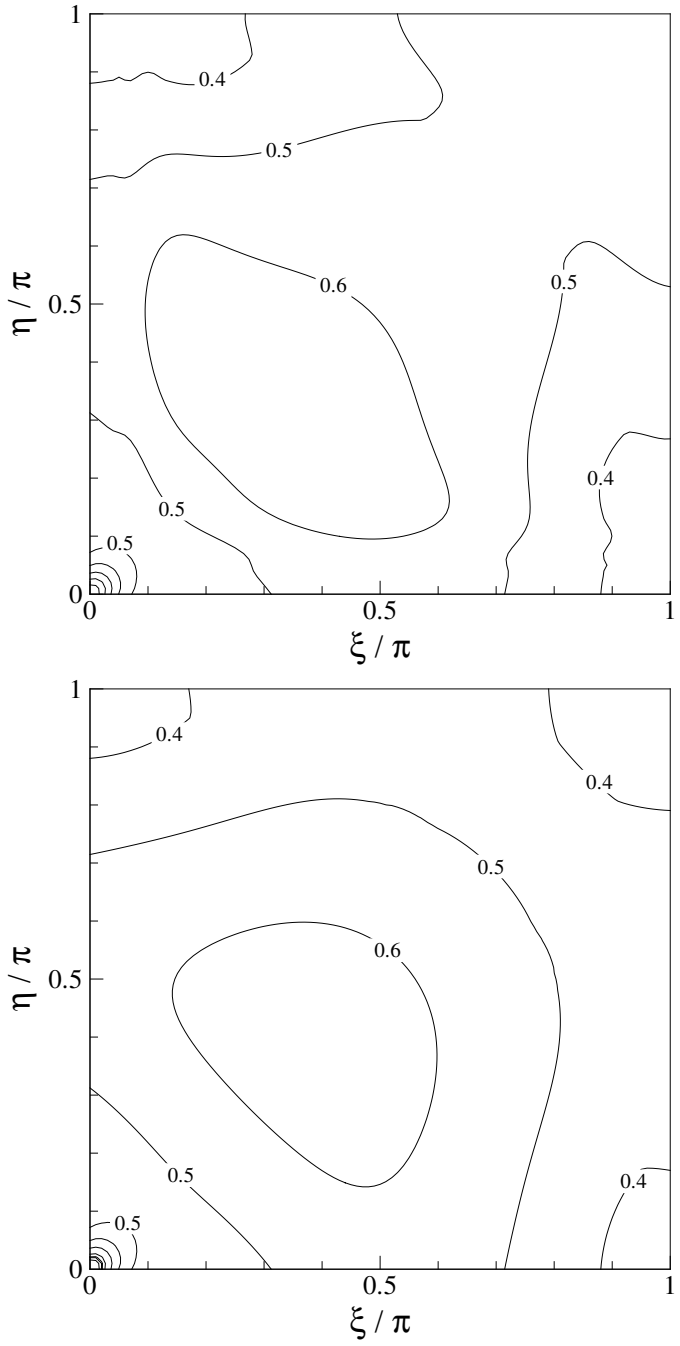


Figure 3.7.3: Amplification factor at $M = 10^{-3}$ and $CFL = 20$. Left : ADI scheme with 20 sub-iterations. Right : Direct Scheme.

3.8 Modified Approximate Factorization Method

The DDADI or MAF method improved by a sub-iterative process aimed at canceling the factorization error can be clearly cast into the general form (3.6.6) using the single implicit operator $\mathcal{H}_\alpha$ $(\alpha = 1)$:

$$\mathcal{H}_{MAF} = (D + \mathcal{S}_1) \cdot D^{-1} \cdot (D + \mathcal{S}_2) \tag{3.8.1}$$

that is

$$\mathcal{H}_{MAF} = \mathcal{H} + \mathcal{S}_1 \cdot D^{-1} \cdot \mathcal{S}_2 \tag{3.8.2}$$

The amplification matrix associated with this method is given by :

$$\begin{cases} G_{MAF} = G_* + V_{MAF}^p \ (Id - G_*) \\ V_{MAF} = (Id - H_{MAF}^{-1} \ H) \end{cases} \tag{3.8.3}$$

with

$$H_{MAF} = (D + S_1) \cdot D^{-1} \cdot (D + S_2) = H + S_1 \cdot D^{-1} \cdot S_2 \tag{3.8.4}$$

As mentioned for the ADI method, one can notice that for one sub-iteration $(p = 1)$, one recovers the original MAF scheme, and therefore the original amplification factor $G_{MAF} = Id - H_{MAF}^{-1} \cdot K$. Let us now study the properties of $MAF(p)$ using Von Neumann analysis.

3.8.1 Global analysis

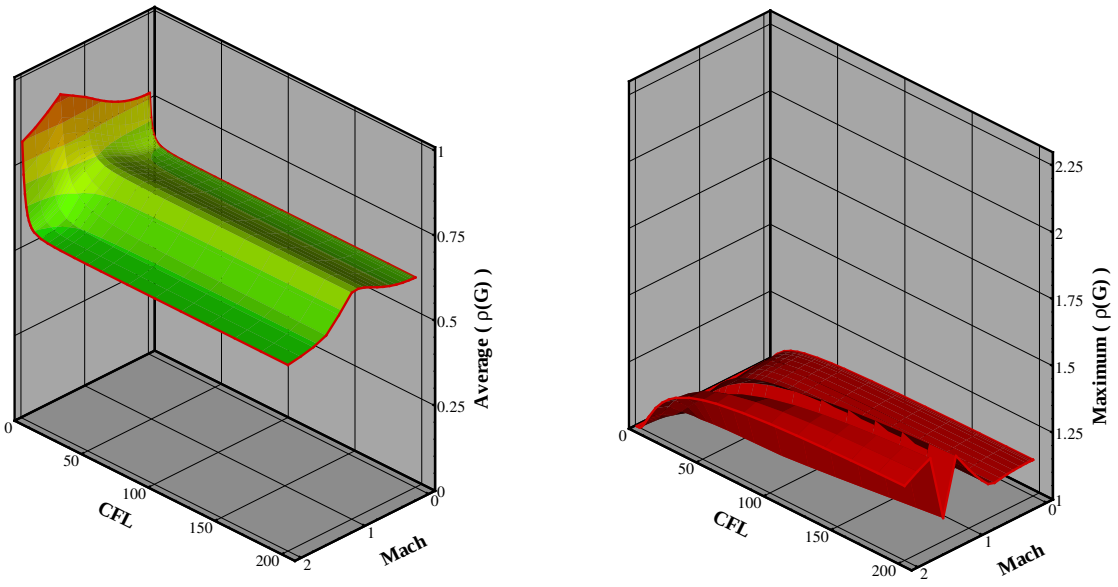


Figure 3.8.1: Average (left) and maximum (right) amplification factor of the non-iterative MAF scheme.

Unlike the ADI scheme, the average amplification factor of the MAF scheme provides a quite good damping for all CFL numbers and for all Mach numbers (*cf.* figure 3.8.1). Unfortunately,

we also note that the maximum value of the amplification factor is greater than unity for the same conditions. It means that the MAF method is unstable if no sub-iteration process is used. The amplification factor of the multi-sweep MAF scheme using 20 sub-iterations is plotted in figure 3.8.2. Using the sub-iteration process improves dramatically the efficiency of the scheme since it recovers the map of the Direct Scheme (*cf.* figure 3.5.3). Besides, as the maximum value of the amplification factor remains equal to unity, the stability is ensured. Hence, the multi-

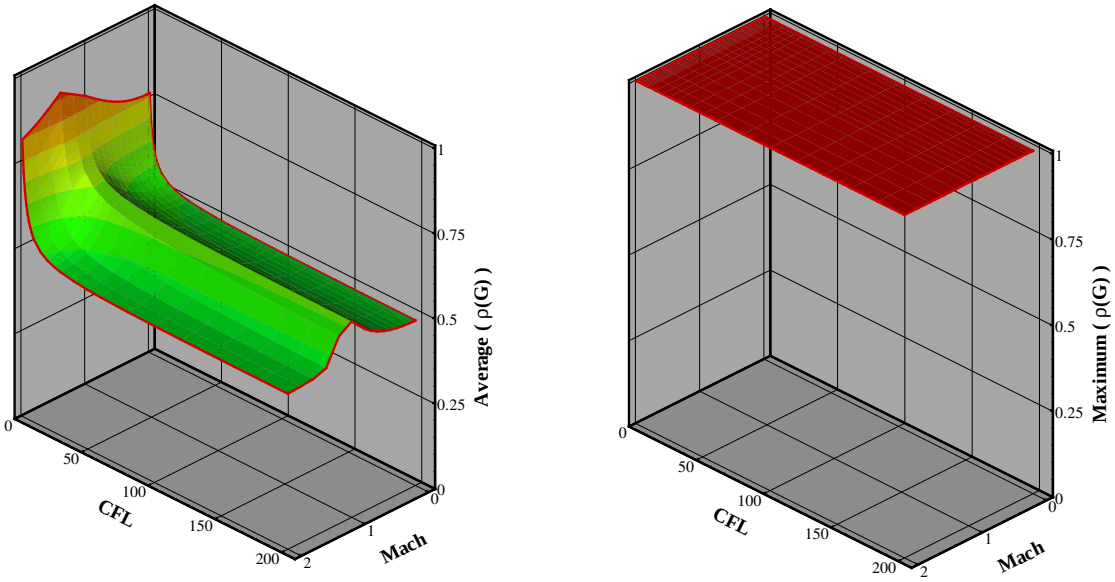


Figure 3.8.2: Average (left) and maximum (right) amplification factor of the multi-sweep MAF scheme with 20 sub-iterations.

sweep MAF scheme is a very competitive method compare to the ADI scheme. This technique enables the use of infinite time-step and it requires a little amount of sweeps to obtain the same efficiency than the Direct Scheme.

3.8.2 Local analysis

Let us now lead a local study of the MAF scheme for a low-Mach case ($M = 10^{-3}$). In figure 3.8.3, we drew the amplification factor of the original MAF scheme at $CFL = 10$ and the one related to the multi-sweep MAF scheme at $CFL = 10^6$ with 20 sub-iterations. As expected from the global analysis, instability arises for the original scheme in the low wavenumber regions. The plot also shows that, in the same wavenumber regions, the damping of such a scheme is quite poor. On the other hand, the multi-sweep scheme provides an excellent damping over the entire wavenumber domain. Nevertheless, compare to the Direct Scheme, we note a little loss of efficiency in the low frequency area. This loss can be circumvented by increasing the number of the sweeps.

Approximate Factorization methods using a sub-iteration process provides very different behaviors according to the choice of Factorization (basic ADI or improved MAF). Indeed, the iterative or multi-sweep strategy reveals itself to be almost useless for the ADI scheme while it dramatically increases the efficiency of the MAF method. Such results have been further

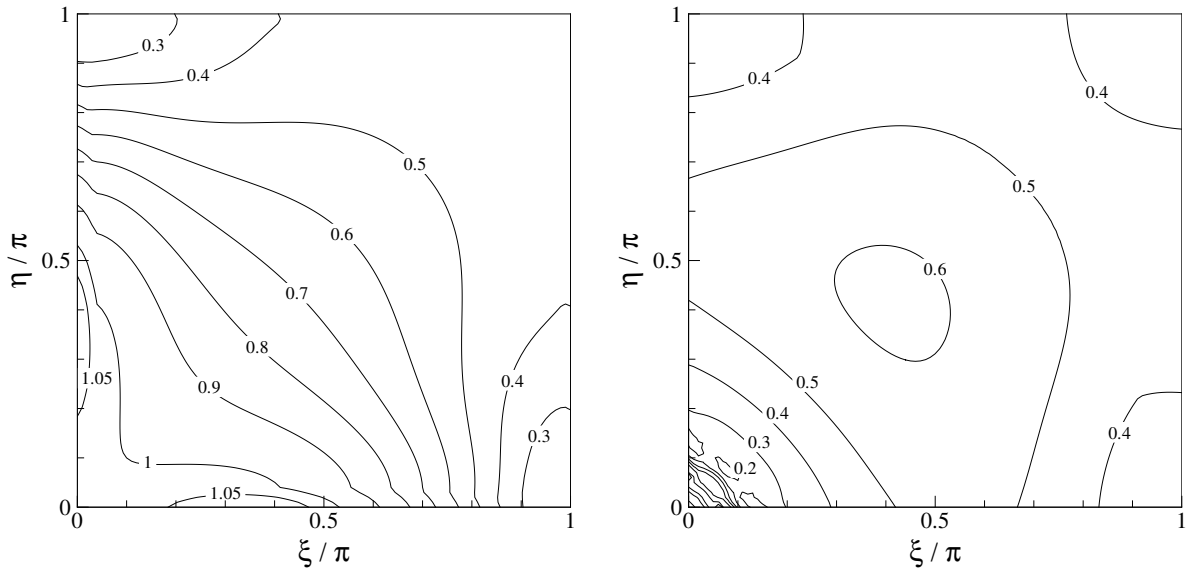


Figure 3.8.3: Amplification factor at $M = 10^{-3}$ and $CFL = 10^{6}$. Left : Original MAF scheme. Right : MAF scheme with 20 sub-iterations.

investigated in [64] and the observations made in this section do not differ from the conclusions previously drawn by these authors.

3.9 Alternate-Line Jacobi relaxation procedure

The ALJ relaxation procedure implies two symmetric sweeps per sub-iteration. Hence, one has to define two operators as follows :

$$\begin{cases} \mathcal{H}_1 = (D + \mathcal{S}_1) \\ \mathcal{H}_2 = (D + \mathcal{S}_2) \end{cases} \tag{3.9.1}$$

The amplification matrix associated with this method is given by :

$$\begin{cases} G_{ALJ} = G_* + V_{ALJ}^p \ (Id - G_*) \\ V_{ALJ} = (Id - H_2^{-1} \ H) \cdot (Id - H_1^{-1} \ H) \end{cases} \tag{3.9.2}$$

with

$$\begin{cases} H_1 = (D + S_1) \\ H_2 = (D + S_2) \end{cases} \tag{3.9.3}$$

It is interesting to point out that in the two-dimensional case, the ALJ scheme is, in fact, identical to the MAF scheme. Indeed, simple algebra gives :

$$V_{ALJ} = Id - (H_2^{-1} \ D \ H_1^{-1}) \cdot H \tag{3.9.4}$$

and obviously we have :

$$H_2^{-1}\ D\ H_1^{-1} = H_{MAF}^{-1} \tag{3.9.5}$$

Consequently, we do not investigate further this method and simply refer to the previous section. However, let us mention $MAF(p)$ and $ALJ(p)$ are no longer identical in the $3D$ case, even though the two methods keep quite similar properties (*cf.* [22] [17]).

3.10 Alternate-Line Gauss-Seidel relaxation procedure

The ALGS relaxation procedure requires four inner-sweeps per sub-iteration (*cf.* chapter 2). It can be cast into the general form (3.6.6) provided 4 operators $\mathcal{H}_\alpha$ are defined as follows :

$$\begin{cases} \mathcal{H}_1 = (D + \mathcal{S}_1 - \dot{B}^+ \mathcal{E}_2^{-1}) \\ \mathcal{H}_2 = (D + \mathcal{S}_1 - \dot{B}^- \mathcal{E}_2^{+1}) \\ \mathcal{H}_3 = (D + \mathcal{S}_2 - \dot{A}^+ \mathcal{E}_1^{-1}) \\ \mathcal{H}_4 = (D + \mathcal{S}_2 - \dot{A}^- \mathcal{E}_1^{+1}) \end{cases} \tag{3.10.1}$$

The amplification matrix associated with this method is then given by :

$$\begin{cases} G_{ALGS} = G_* + V_{ALGS}^p\ (Id - G_*) \\ V_{ALGS} = (Id - H_4^{-1}\ H) \cdot (Id - H_3^{-1}\ H) \cdot (Id - H_2^{-1}\ H) \cdot (Id - H_1^{-1}\ H) \end{cases} \tag{3.10.2}$$

with the Fourier symbols :

$$\begin{cases} H_1 = (D + S_1 - \dot{B}^+ e^{-\underline{\mathrm{i}}\eta}) \\ H_2 = (D + S_1 - \dot{B}^- e^{+\underline{\mathrm{i}}\eta}) \\ H_3 = (D + S_2 - \dot{A}^+ e^{-\underline{\mathrm{i}}\xi}) \\ H_4 = (D + S_2 - \dot{A}^- e^{+\underline{\mathrm{i}}\xi}) \end{cases} \tag{3.10.3}$$

Let us now study the efficiency of this scheme using Von Neumann analysis.

3.10.1 Global analysis

The average $\bar{\rho}_{ALGS}$ and maximum ρ_{ALGS}^{max} amplification factor of the ALGS method are plotted in figure 3.10.1. A strong instability is noticeable for the highest CFL numbers. Even though this scheme provides a very good global damping for all CFL numbers and all Mach number, it cannot be used for CFL greater than 30. Increasing the number of sweeps do not circumvent these instabilities, on the contrary it deteriorates even more dramatically the amplification factor (*cf.* figure 3.10.2). The formula (3.6.8) provides a clear explanation of this behavior, at least when a scalar advection problem is consider - but the mechanism extends to the system case - : the relationship $|G - G_*| = |V|^p |1 - G_*|$ can be immediately deduced from (3.6.8); if $|V|^p > 1$,

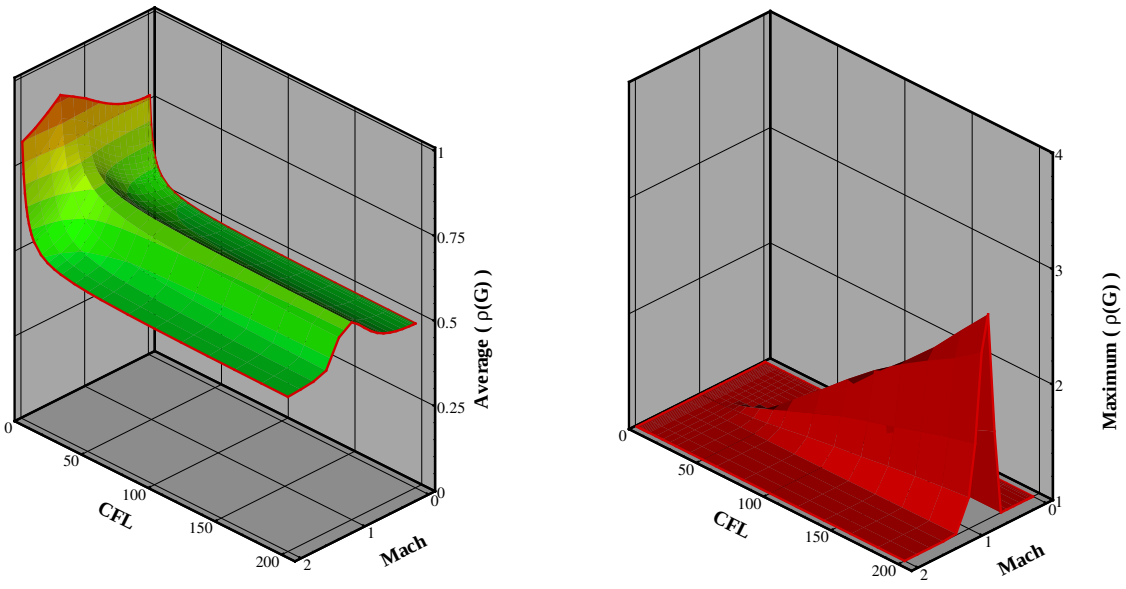


Figure 3.10.1: Average (left) and maximum (right) amplification factor of the ALGS scheme with one sub-iteration.

which turns to be the case when ALGS is applied to the first-order upwind implicit stage, then increasing the number of sub-iterations p tends to further increase the difference between G_* and G up to a complete divergence of the iterative solver. Such a behaviour was outlined by Buelow *et al.* in [12]. Analyzes of the I/I ALGS scheme performed in this work also show unstable behaviors for high CFL numbers. This means that this instability does not arise from the non-consistent choice of a first-order implicit stage / third-order explicit stage but is directly related to the first-order implicit stage. This point is further confirmed by stability results obtained with ALGS applied to a Lax-Wendroff type implicit scheme for which no instability is detected [18]. In fact, this instability of $ALGS(p)$ seems to be closely connected with the multi-dimensional upwind implicit stage : ALGS inner iteration process can be split into two main sweeps, the first one in the x-direction, the second one in the y-direction; now, using a sweep in one direction only avoids the occurrence of instabilities but leads to a poor damping of the error while the combination of both sweeps can provide a very good damping if the time-step is restricted.

3.10.2 Local analysis

In figure 3.10.3, the amplification factor of the ALGS(1) scheme is presented for $M = 10^{-3}$ and for two different CFL numbers. As previously shown on the global analysis, for $CFL = 10^6$, there exists a little peak for which the amplification factor is greater than unity. This peak is located in the very low frequency area, consequently, in practical computations, if the mesh is not fine enough, high CFL numbers can be used anyhow without triggering instability. This may be the reason for the conventional wisdom which considers Gauss-Seidel method applied to a block-implicit upwind scheme as unconditionally stable. On the other hand, the ALGS(1) scheme at $CFL = 20$ is as efficient as the Direct Scheme (*cf.* figure 3.7.3) and remains stable over the entire wavenumber domain. Thus, unless the time-step has to be restricted, the ALGS(1) scheme provides a good damping, moreover it requires only one sub-iteration which is very cheap

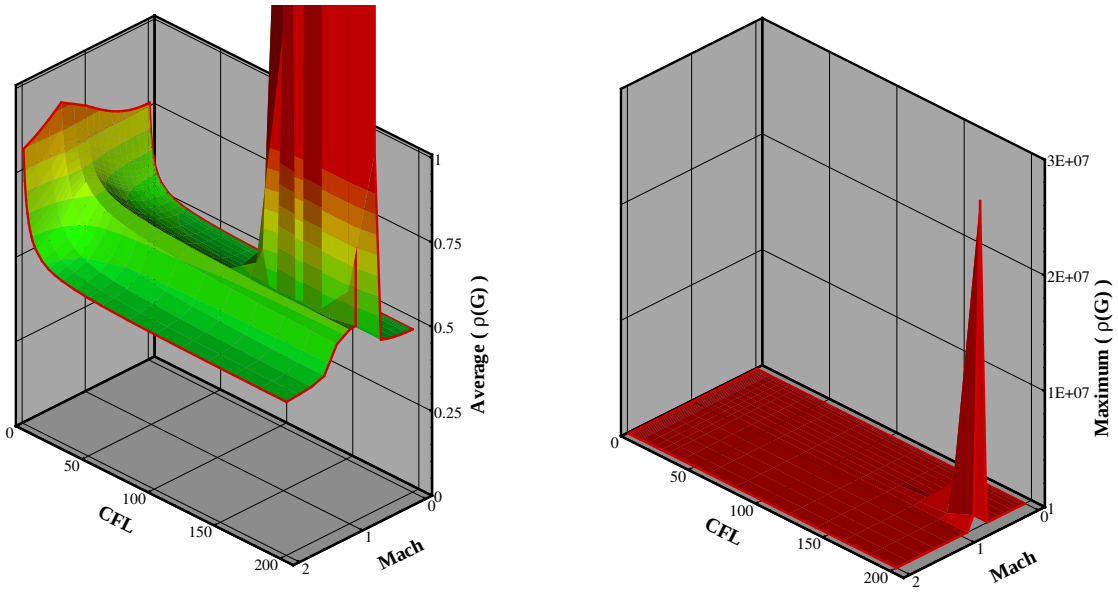


Figure 3.10.2: Average (left) and maximum (right) amplification factor of the ALGS scheme with 20 sub-iterations.

for practical computations.

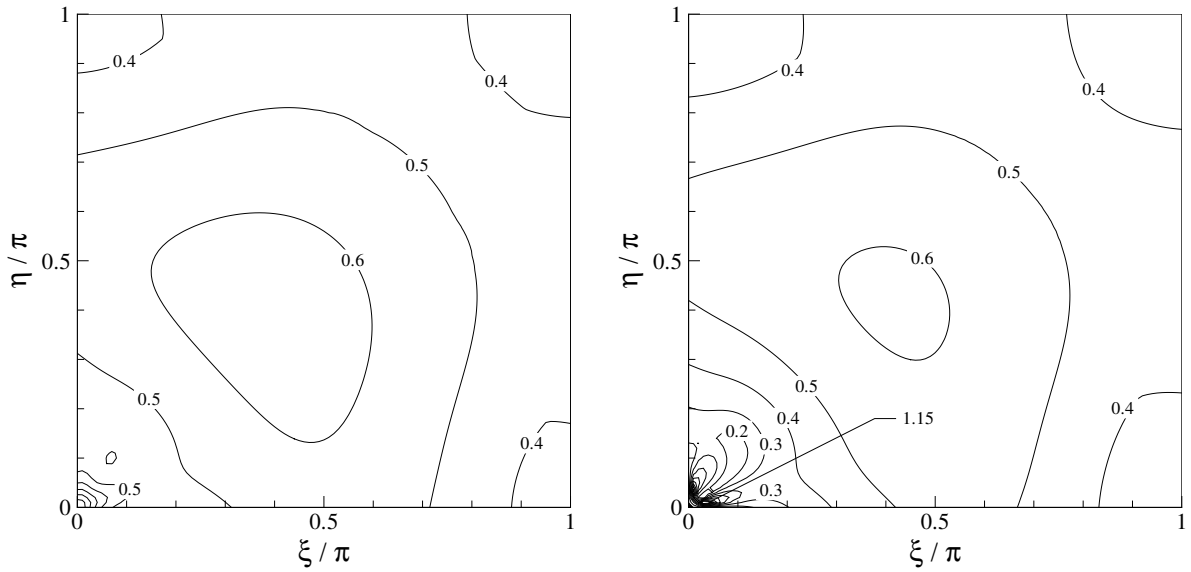


Figure 3.10.3: Amplification factor of the ALGS(1) scheme at $M = 10^{-3}$. Left : $CFL = 20$. Right : $CFL = 10^6$.

3.11 Point Jacobi relaxation procedure

PJ scheme is certainly the simplest relaxation method. Indeed the associated single operator $\mathcal{H}_\alpha$ ($\alpha = 1$) is given by :

$$\mathcal{H}_{PJ} = D \tag{3.11.1}$$

The amplification matrix is therefore defined as follows :

$$\begin{cases} G_{PJ} = G_* + V_{PJ}^p \ (Id - G_*) \\ V_{PJ} = (Id - D^{-1} \ H) \end{cases} \tag{3.11.2}$$

Unfortunately, such a simplicity involves an important lack of efficiency which will appear in the Von Neumann analysis described in the next paragraph.

3.11.1 Global analysis

The PJ scheme without inner iteration ($p = 1$) is strongly unstable for all CFL numbers and for all Mach numbers (*cf.* figure 3.11.1). This instability is partially circumvented if 50 sub-iterations are made : $PJ(50)$ scheme is then stable for CFL numbers smaller than 30. The average of the amplification factor seems to show a rather good damping in such conditions. Let us now make a local analysis in order to investigate a bit more deeply the particular behavior of the $PJ(p)$ scheme.

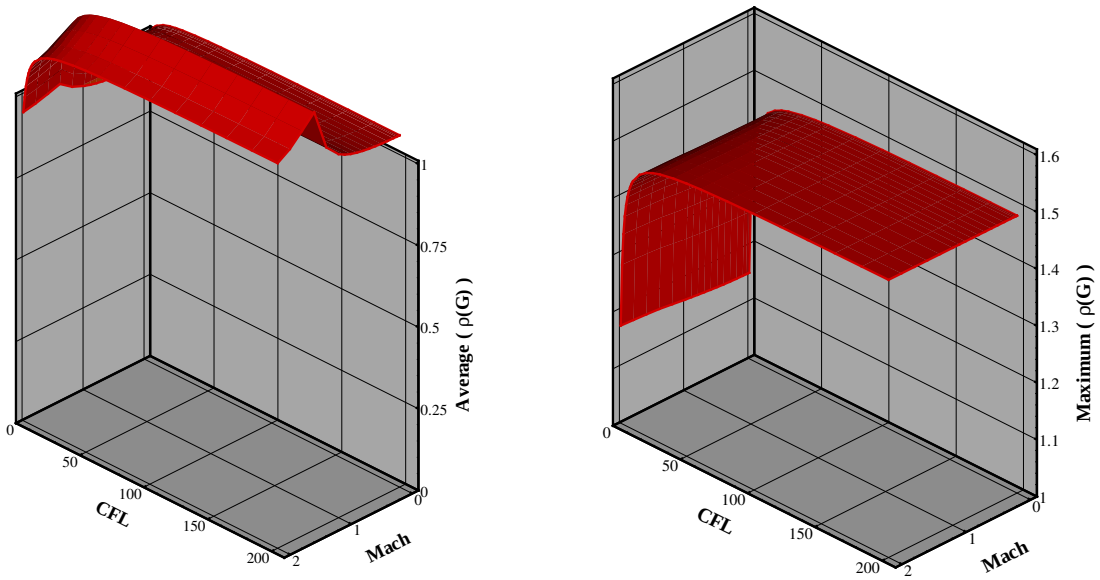


Figure 3.11.1: Average (left) and maximum (right) amplification factor of the PJ(1) scheme.

3.11.2 Local analysis

The local analysis is performed for $M = 10^{-3}$ and $CFL = 20$ (stable conditions) and results are presented in figure 3.11.3. Using PJ scheme with 50 sub-iterations enables to obtain a good

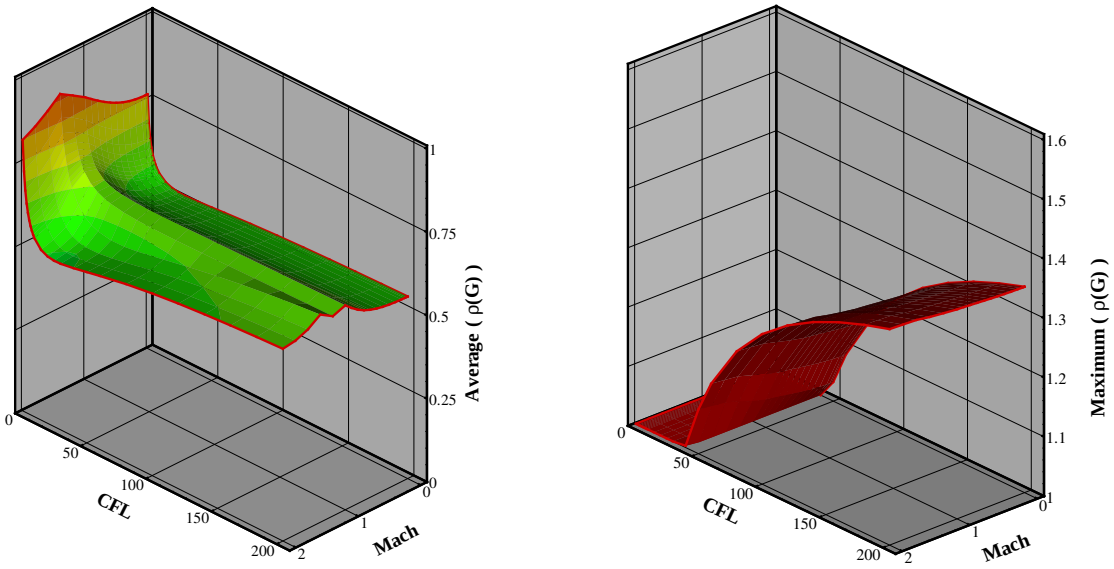


Figure 3.11.2: Average (left) and maximum (right) amplification factor of the PJ scheme with 50 sub-iterations.

damping over a large part of the wavenumber domain. The amplification factor map becomes close to the Direct Scheme's one (*cf.* figure 3.11.3); however, spurious oscillations occur, specially for the high frequencies, along the diagonal line $\xi = \eta$. To emphasize this behavior a cutline of the amplification factor along $\xi = \eta$ is also plotted for the PJ scheme and the Direct Scheme. Reducing the number of sub-iterations yields less oscillations but of larger amplitude which can therefore leads to instabilities. On the other hand, increasing the number of sub-iterations yields a greater number of oscillations, each of smaller amplitude. Using 100 sub-iterations at $CFL = 20$ allows to make all the oscillations vanish or, more precisely, yields such small-amplitude oscillations that they can no longer be detected. During the course of this study, it was noted that the Line-Jacobi scheme displays a similar behavior - note Line-Jacobi (LJ) means relaxation is performed along the x direction or the y direction and not successively along both directions as in the Alternate-Line Jacobi method. The LJ method applied in the first (x) space-direction makes use of the operator $\mathcal{H}_1 = D + \mathcal{S}_1$ and its associated V_{LJ} coefficient reads :

$$V_{LJ} = Id - H_1^{-1}\ H = -(D + S_1)^{-1} S_2 \tag{3.11.3}$$

The amplification map of this method is presented in figure 3.11.4. The efficiency is clearly close to the Direct Scheme's one but we note that here again oscillations appear, along the line $\xi = 0$ in this case. The plot of the amplification factor along this line shows a "sine curve" phenomenon. For both the PJ and LJ methods, this oscillation problem highly disturbs the damping at high CFL numbers since a large amount of sub-iterations is needed to cancel the "sine curve" effect, making these methods potentially expensive.

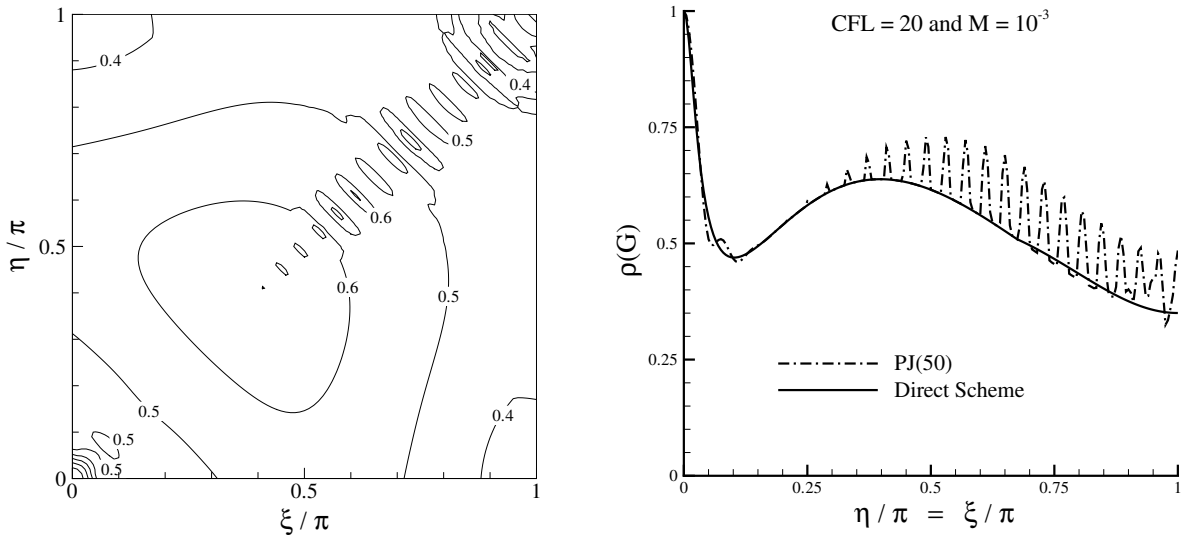


Figure 3.11.3: Amplification factor of the PJ(50) scheme at $M = 10^{-3}$ and $CFL = 20$. Left : Map over the whole wavenumber domain. Right : Diagonal profile along $\xi = \eta$.

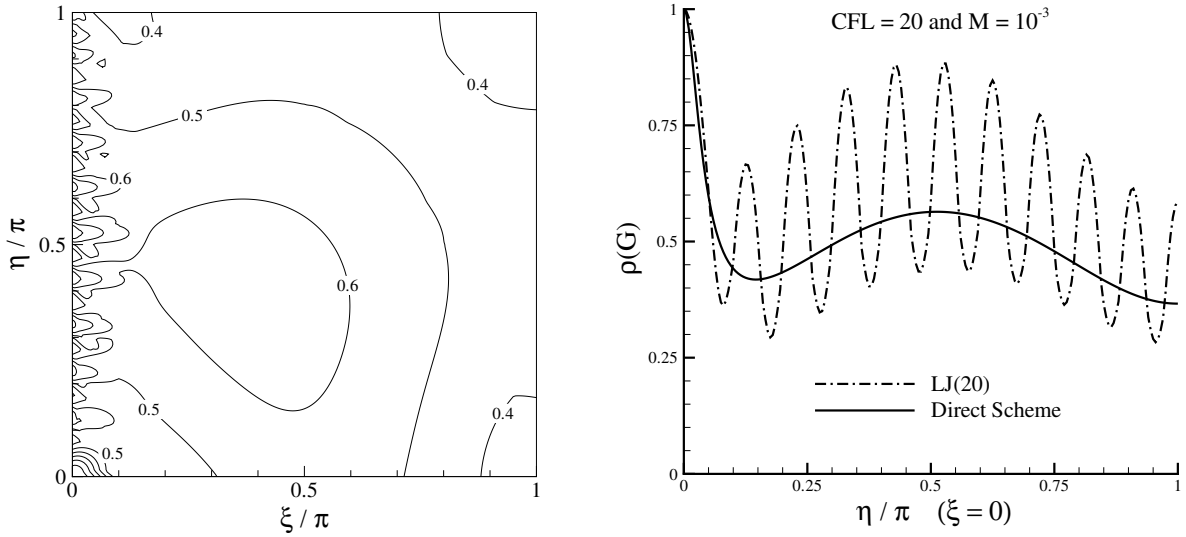


Figure 3.11.4: Amplification factor of the LJ(20) scheme at $M = 10^{-3}$ and $CFL = 20$. Left : Map over the whole wavenumber domain. Right : Diagonal profile along $\xi = \eta$.

3.12 Symmetric Gauss-Seidel relaxation procedure

The SGS relaxation procedure requires two inner-sweeps per sub-iteration (*cf.* chapter 2); these two steps are defined by the two following operators as follows :

$$\begin{cases} \mathcal{H}_1 = (D - \dot{A}^{+}\mathcal{E}_1^{-1} - \dot{B}^{+}\mathcal{E}_2^{-1}) \\ \mathcal{H}_2 = (D + \dot{A}^{-}\mathcal{E}_1^{+1} + \dot{B}^{-}\mathcal{E}_2^{+1}) \end{cases} \tag{3.12.1}$$

The amplification matrix associated with this method is then given by :

$$\begin{cases} G_{SGS} = G_* + V_{SGS}^p \ (Id - G_*) \\ V_{SGS} = (Id - H_2^{-1} \ H) \cdot (Id - H_1^{-1} \ H) \end{cases} \tag{3.12.2}$$

with

$$\begin{cases} H_1 = (D - \dot{A}^+ e^{-\underline{i}\xi} - \dot{B}^+ e^{-\underline{i}\eta}) \\ H_2 = (D + \dot{A}^- e^{+\underline{i}\xi} + \dot{B}^- e^{+\underline{i}\eta}) \end{cases} \tag{3.12.3}$$

When only one sub-iteration is performed, SGS scheme is also called LU approximate factorization method, and will be denoted LU-SGS in the remainder of the section (rather than $SGS(1)$). Now, let us study the efficiency of LU-SGS and $SGS(p)$ using the Von Neumann analysis.

3.12.1 Global analysis

The maximum and the average values of the amplification factor versus CFL and Mach numbers are presented in figures 3.12.1 and 3.12.2. We notice that LU-SGS scheme is unstable for Mach numbers close to unity, even if the CFL is small. It is therefore necessary to perform several inner iterations in order to circumvent the instability. As shown in figure 3.12.2, using 20 sub-iterations enables to recover the efficiency of the Direct Scheme. It is noticeable that, unlike the ALGS scheme, the SGS scheme remains stable even at high CFL numbers. However, Buelow *et al.* in [12] show that SGS(p) loses a great part of its efficiency for high aspect ratios - this point will be further investigated in the second part of this thesis, when SGS is applied for solving a matrix-free implicit stage. Nevertheless, the SGS method remains of great interest since it can be easily applied on unstructured grids, like PJ of course, which is not the case for other methods.

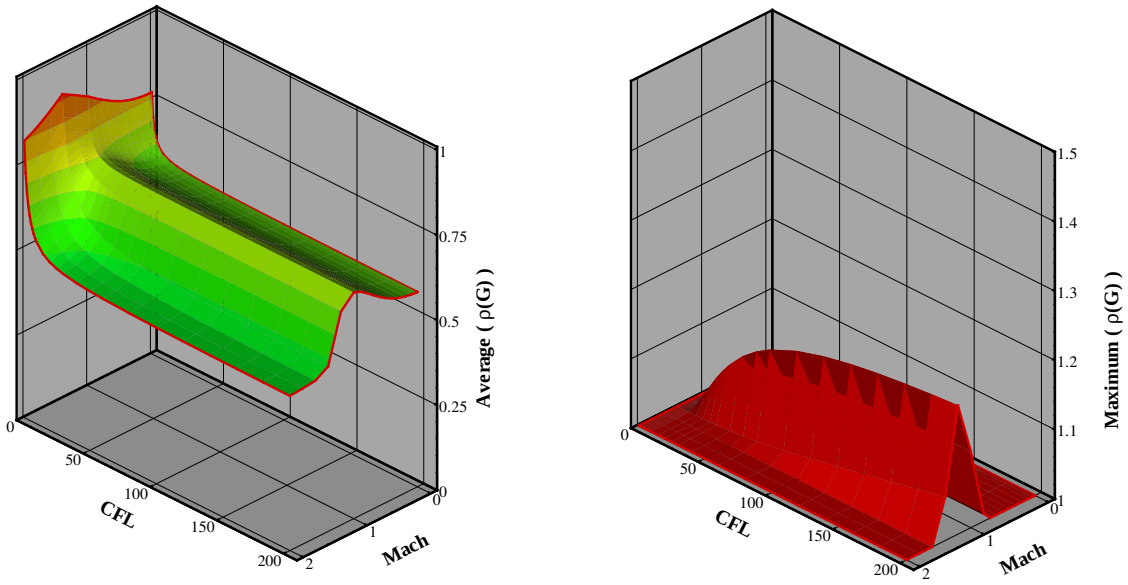


Figure 3.12.1: Average (left) and maximum (right) amplification factor of the LU-SGS scheme.

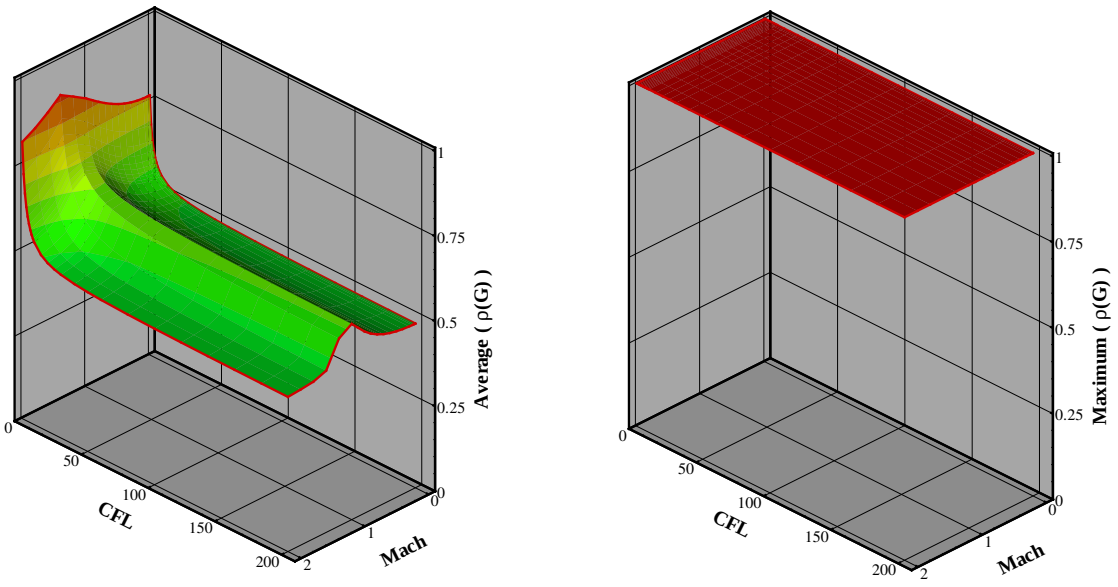


Figure 3.12.2: Average (left) and maximum (right) amplification factor of the SGS scheme with 20 sub-iterations.

3.12.2 Local analysis

The local analysis is made for a low-Mach number case ($M = 10^{-3}$) and at $CFL = 10^6$. In such conditions, the LU-SGS scheme is stable but its efficiency is poor, especially in the low wavenumber regions (*cf.* figure 3.12.3). Using 20 inner-iterations provides excellent damping over the entire wavenumber domain, the efficiency is very close to the Direct Scheme's one making the SGS scheme an attractive method, in particular for computing flows on unstructured grids, since it can be easily implemented on such grids.

3.13 Conclusions

The present chapter concludes the first part of this thesis devoted to the analysis of some classical implicit methods, with the objective to select the most efficient treatment for the block implicit scheme with which we plan to compare the matrix-free treatment developed in the next part of the thesis. Let us recall memory storage has not yet been taken into account in our analysis of efficiency, which has been mainly focused on intrinsic efficiency. Among the line implicit treatments, *i.e.* those that make use at some stage of tridiagonal system inversion(s), ALGS(p) with p very small (typically $p = 1$ or 2) appears as the most efficient treatment, even though some instabilities may occur on very fine grids. The point implicit treatments such as PJ(p) and SGS(p) offer a much lower unit cost since they only require the local inversion of a 4×4 (in $2D$) block for each non-linear iteration but since they also need a larger number of inner iterations (especially PJ(p) because of the oscillatory behavior of the amplification factor) to achieve the Direct Solver's intrinsic efficiency they are not necessarily more globally efficient than ALGS(1) for instance. The hierarchy between ALGS(p) and mostly SGS(p) depends also on the unit cost of each treatment, itself dependent on the particular implementation of the method within

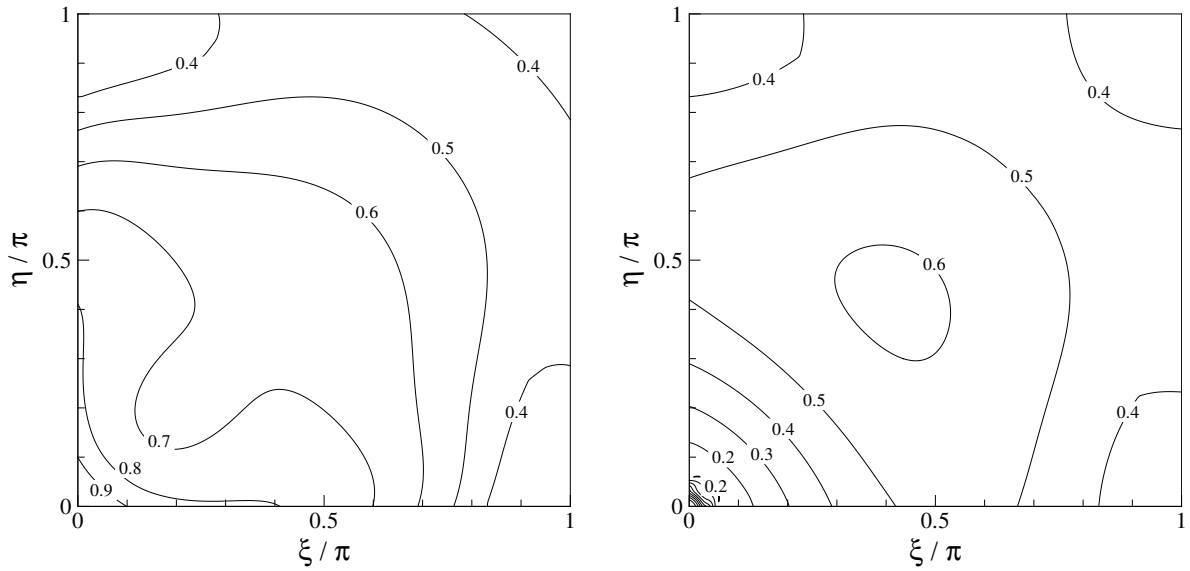


Figure 3.12.3: Amplification factor of the SGS scheme at $M = 10^{-3}$ and $CFL = 10^{6}$. Left : SGS(1). Right : SGS(20).

the flow solver. In this work, all the block implicit treatments presented up to now have been implemented within a "model" structured flow solver, dedicated to the solution of the Euler and Navier-Stokes equations on Cartesian grids. Numerical experiments carried out with this code have shown ALGS(1) was in practice the most efficient treatment (regardless of memory issue); consequently, ALGS(1) will be retained in the second part of this work as the reference block implicit treatment (as far as computational efficiency is concerned). It is worthwhile to point out SGS(p), though found less computationally efficient than ALGS(1) when used in our model structured-grid flow solver to solve the block-implicit Roe upwind scheme, is particularly interesting in the perspective of an application in the context of unstructured grids. With this property in mind, a particular attention will be devoted to this point treatment in the next part of this thesis when looking for an approximate solution of the matrix-free implicit stage since the planned implementation of the matrix-free treatment in CAST3M has to be realized for unstructured grid computations (note also the PJ(p) will be reconsidered because its properties when applied to the matrix-free implicit stage will be quite different from those observed with the block implicit stage).

Part II

Low-Cost Efficient Implicit Treatments

Chapter 4

Matrix-Free Implicit Method

4.1 Objectives

The first part of the thesis has been dedicated to the description of implicit treatments which provide a high intrinsic efficiency, namely a fast damping of all error modes. However, as mentioned in the introduction of this work, such an efficiency is usually obtained at the expense of a relatively high unit computational cost and strong memory requirements. The excessive memory demand introduced by an implicit stage is mainly related to the use of block Jacobian matrices when linearizing the numerical fluxes so as to derive the implicit part of the scheme. Hence, a way to reduce memory requirements is to develop implicit treatments that do not rely on full Jacobian matrices and this may be achieved by introducing simplifications in the explicit stage linearization. The price to pay for such a simplified choice of implicit stage is a loss of intrinsic efficiency; nevertheless, if the unit computational cost is also reduced, a competitive global efficiency can be preserved. As already pointed out in the general introduction of this work, a low-cost/low-memory/low intrinsic efficiency implicit treatment is one of the trade-off solutions to the multi-objective optimization problem represented by the derivation of a globally efficient implicit scheme. The second part of this thesis is entirely devoted to the review of

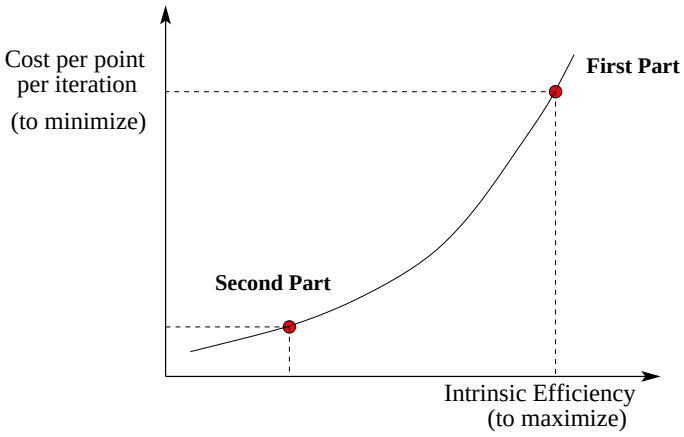


Figure 4.1.1: Pareto optimal set for an efficient implicit treatment.

such low-cost treatments as alternative solutions to the high-cost/large-memory/high intrinsic efficiency treatments detailed in the first part of the thesis (see also figure 4.1).
Several low-cost implicit treatments have been developed since the early-eighties. The diagonalized implicit treatments firstly introduced by Pulliam and Chaussée [65] in the context of compressible flows and recently extended to low-Mach unsteady flows [10] allow to reduce unit cost but they do not ensure enough reduction in memory requirements when large scale problems are computed. It is therefore necessary to opt for even more simplified methods such as Matrix-Free implicit treatments. These methods consist in replacing the absolute values of Jacobian matrices that appear in the implicit stage after the linearization of an upwind scheme, by their spectral radius [38]. Then, approximating the product between Jacobian matrix and time-increments of the conserved variables by the time-increment of the flux-vectors [71], it is possible to obtain a truly Matrix-Free implicit scheme [48]. However, when low-Mach unsteady flows are computed using a dual-time stepping approach, the scheme loses its matrix-free properties and both unit cost and memory requirements increase significantly. The objective of this chapter is to outline an implicit treatment which preserves the simplicity of the Matrix-Free approach for low-Mach unsteady flows. The difficulties related to this kind of flows will be detailed and it will be shown how to reap the benefit of the Turkel's preconditioner's properties so as to produce a truly Matrix-Free scheme for all speed flows. The following chapters (5, 6 and 7) will then be devoted to the systematic analysis of the intrinsic efficiency of the new implicit treatment for a wide range of configurations (inviscid or viscous, steady or unsteady flows); in particular, the *a priori* Von Neumann analysis will give us the opportunity to tune properly the cut-off definition that must be selected when computing low-Mach number flows with a preconditioned upwind scheme : the need to adapt this definition to the type of problem under study (inviscid/viscous, steady/unsteady) will be underlined. The last chapter of this second part will eventually provide some comparisons between an efficient implicit block treatment and the Matrix-Free schemes over representative test-cases, computed on structured grids. The possibility to obtain a satisfactory global efficiency from a low-cost matrix-free treatment will be demonstrated in a way convincing enough to justify further developments for the implementation of this technique in the numerical platform CAST3M.

4.2 Matrix-Free implicit treatment

4.2.1 General principles

Let us first describe the design principles of the Matrix-Free implicit treatment for the 2D Euler equations. In the manner of the first chapter, let us consider the class of upwind schemes which are defined as follows :

$$F^n_{i+\frac{1}{2},j} = \left((\mu_1 f^E)^n - D_1\delta_1 w^n\right)_{i+\frac{1}{2},j}$$

$$G^n_{i,j+\frac{1}{2}} = \left((\mu_2 g^E)^n - D_2\delta_2 w^n\right)_{i,j+\frac{1}{2}}$$

where D_1 and D_2 are matrices of numerical dissipation. The implicit stage reads :

$$\frac{w^{n+1}_{i,j} - w^n_{i,j}}{\Delta t} + \left(\frac{\delta_1 F}{\delta x} + \frac{\delta_2 G}{\delta y}\right)^{n+1}_{i,j} = 0 \tag{4.2.1}$$

The derivation of a Matrix-Free implicit stage is carried out in two steps. First, only the dissipative parts of the numerical fluxes are linearized around time level $(n+1)$:

$$\begin{aligned} [D_1\delta_1 w]^{n+1} &= D_1^n\delta_1 w^n + \Delta\,[D_1\delta_1 w]^n \approx D_1^n\delta_1 w^n + D_1^n\delta_1\Delta w^n \ , \\ [D_2\delta_2 w]^{n+1} &= D_2^n\delta_2 w^n + \Delta\,[D_2\delta_2 w]^n \approx D_2^n\delta_2 w^n + D_2^n\delta_2\Delta w^n \ , \end{aligned} \tag{4.2.2}$$

while the centered parts of the numerical fluxes are computed using the definition of the time increment $\Delta\phi^n = \phi^{n+1} - \phi^n$:

$$\mu_1(f^E)^{n+1} = \mu_1(f^E)^n + \mu_1\Delta(f^E)^n \quad , \qquad \mu_2(g^E)^{n+1} = \mu_2(g^E)^n + \mu_2\Delta(g^E)^n \quad . \tag{4.2.3}$$

Hence, the numerical fluxes at time level $(n+1)$ now read :

$$\begin{aligned} F^{n+1} &\approx F^n + \mu_1\Delta(f^E)^n - D_1^n\delta_1\Delta w^n \ , \\ G^{n+1} &\approx G^n + \mu_2\Delta(g^E)^n - D_2^n\delta_2\Delta w^n \ . \end{aligned} \tag{4.2.4}$$

The second step consists in replacing the dissipation matrices D_1 and D_2 by the numerical scalar dissipation of the Rusanov scheme :

$$\begin{aligned} D_1 &\longrightarrow \tfrac{1}{2}\rho(A^E)Id = \tfrac{1}{2}\rho_1^E Id \\ D_2 &\longrightarrow \tfrac{1}{2}\rho(B^E)Id = \tfrac{1}{2}\rho_2^E Id \end{aligned} \tag{4.2.5}$$

Note such a simplification can be performed without altering the stability of the numerical treatment because the dissipation matrices are definite positive. The numerical fluxes at time level $(n+1)$ are finally approximated as follows :

$$\begin{aligned} F^{n+1} &\approx F^n + \mu_1\Delta(f^E)^n - \tfrac{1}{2}(\rho_1^E)^n\delta_1\Delta w^n \ , \\ G^{n+1} &\approx G^n + \mu_2\Delta(g^E)^n - \tfrac{1}{2}(\rho_2^E)^n\delta_2\Delta w^n \ . \end{aligned} \tag{4.2.6}$$

Inserting expressions (4.2.6) into the implicit stage (4.2.1) yields :

$$\begin{aligned} C_0\Delta w_{i,j}^n \ + \ C_1^-\Delta w_{i-1,j}^n + \ C_1^+\Delta w_{i+1,j}^n + \ C_2^-\Delta w_{i,j-1}^n + \ C_2^+\Delta w_{i,j+1}^n & \\ = \ -\Delta t\cdot\mathcal{R}_{i,j}^n \ - \ \sigma_1\delta_1\mu_1\Delta(f^E)_{i,j}^n \ - \ \sigma_2\delta_2\mu_2\Delta(g^E)_{i,j}^n & \end{aligned} \tag{4.2.7}$$

where the explicit residual $\mathcal{R}_{i,j}^n$ is defined as follows :

$$\mathcal{R}_{i,j}^n = \left(\frac{\delta_1 F}{\delta x} + \frac{\delta_2 G}{\delta y}\right)_{i,j}^n \ ,$$

and where the coefficients C are now given by :

$$\left\{\begin{array}{l} C_1^- = -\frac{1}{2}\sigma_1\left(\rho_1^E\right)_{i-\frac{1}{2},j}^n \\ C_1^+ = -\frac{1}{2}\sigma_1\left(\rho_1^E\right)_{i+\frac{1}{2},j}^n \\ C_2^- = -\frac{1}{2}\sigma_2\left(\rho_2^E\right)_{i,j-\frac{1}{2}}^n \\ C_2^+ = -\frac{1}{2}\sigma_2\left(\rho_2^E\right)_{i,j+\frac{1}{2}}^n \\ C_0 \ = 1 - C_1^- - C_1^+ - C_2^- - C_2^+ \end{array}\right. \tag{4.2.8}$$

Eventually, the implicit contributions in the LHS and RHS of scheme (4.2.7) are truly Matrix-Free and it is noticeable that the simplifications that have been performed do not affect the explicit stage accuracy. Thus, the accuracy of the scheme remains unchanged provided that the method converges to a steady-state. Moreover, let us emphasize that this Matrix-Free implicit treatment does only require simplifications and approximations over the dissipative terms derived from the linearization. The scheme (4.2.7) involving increment vector Δw^n and flux increment vectors $\Delta(f^E)^n$ and $\Delta(g^E)^n$ has now to be solved.

Remark :

A number of viewpoints are encountered in the literature when it comes to the derivation of the previous matrix-free implicit stage; they are briefly reviewed here for the sake of a complete reference to existing works. Following the viewpoint adopted by Sharov and Nakahashi [71] and later used by Löhner *et al.* in [48], a starting point for the matrix-free treatment derivation will be the classical linearization of the numerical fluxes around time level $(n+1)$:

$$F^{n+1} \approx F^n + (A^E)^n \mu_1 \Delta w^n - D_1^n \delta_1 \Delta w^n \ ,$$

$$G^{n+1} \approx G^n + (B^E)^n \mu_2 \Delta w^n - D_2^n \delta_2 \Delta w^n \ .$$

Then, performing a spectral radius simplification over the dissipative terms leads to :

$$F^{n+1} \approx F^n + (A^E)^n \mu_1 \Delta w^n - \tfrac{1}{2}(\rho_1^E)^n \delta_1 \Delta w^n \ ,$$

$$G^{n+1} \approx G^n + (B^E)^n \mu_2 \Delta w^n - \tfrac{1}{2}(\rho_2^E)^n \delta_2 \Delta w^n \ .$$

Inserting these expressions into the implicit stage yields :

$$\begin{aligned} &\Delta w_{i,j}^n \ + \ \sigma_1\delta_1((A^E)^n\mu_1\Delta w^n)_{i,j} \ - \ \tfrac{1}{2}\sigma_1\delta_1\left((\rho_1^E)^n\delta_1\Delta w^n\right)_{i,j} \\ &\qquad + \sigma_2\delta_2((B^E)^n\mu_2\Delta w^n)_{i,j} \ - \ \tfrac{1}{2}\sigma_2\delta_2\left((\rho_2^E)^n\delta_2\Delta w^n\right)_{i,j} \ = \ -\Delta t\cdot\mathcal{R}_{i,j}^n \end{aligned} \tag{4.2.9}$$

At this point, it is noticeable that, although a spectral radius simplification was introduced, the LHS still contains Jacobian matrices A^E and B^E. Hence, the resolution of the system remains expensive and the issues related to memory requirements are not solved. In order to circumvent this difficulty, the product of the Jacobian matrix and incremental vector can be approximated by the increment of the flux vector :

$$\begin{aligned} A^E\cdot\mu_1\Delta w^n &\approx \mu_1\Delta(f^E)^n = \mu_1\left((f^E)^{n+1} - (f^E)^n\right) \\ B^E\cdot\mu_2\Delta w^n &\approx \mu_2\Delta(g^E)^n = \mu_2\left((g^E)^{n+1} - (g^E)^n\right) \end{aligned} \tag{4.2.10}$$

Inserting these relations in (4.2.9) yields :

$$\begin{aligned} C_0\Delta w_{i,j}^n \ + \ C_1^-\Delta w_{i-1,j}^n + \ C_1^+\Delta w_{i+1,j}^n + \ C_2^-\Delta w_{i,j-1}^n + \ C_2^+\Delta w_{i,j+1}^n & \\ = \ -\Delta t\cdot\mathcal{R}_{i,j}^n \ - \ \sigma_1\delta_1\mu_1\Delta(f^E)_{i,j}^n \ - \ \sigma_2\delta_2\mu_2\Delta(g^E)_{i,j}^n & \end{aligned}$$

where (4.2.7) is recovered, with the coefficients $C_p^{\pm}$ still defined by (4.2.8).

Since some numerical fluxes such as AUSM+ or CUSP cannot be easily written under the form of a centered flux augmented with a matrix or scalar dissipation flux, yet another viewpoint for the design of the Matrix-Free implicit scheme can be given. Let us indeed consider the numerical fluxes F^{n+1} and G^{n+1} and let us expand these fluxes as follows :

$$F^{n+1} = F^n + \Delta F^n \ ,$$

$$G^{n+1} = G^n + \Delta G^n \ .$$

In order to obtain a Matrix-Free implicit stage, an inconsistent evaluation of the time-increment flux vectors is performed, *i.e.* the latter are computed using the first order space accurate Rusanov scheme :

$$\Delta F^{Rus} = \mu_1(\Delta(f^E)^n) - \tfrac{1}{2}\rho_1^E\delta_1(\Delta w^n) \ ,$$

$$\Delta G^{Rus} = \mu_2(\Delta(g^E)^n) - \tfrac{1}{2}\rho_2^E\delta_2(\Delta w^n) \ .$$

We recall that once the steady solution is reached, these increments vanish and, therefore, the choice of the Rusanov scheme to evaluate them does not affect the accuracy of the solution. Now inserting these expressions into the implicit stage yields again the previously obtained matrix-free implicit stage :

$$\begin{aligned}\Delta w_{i,j}^n \ + \ \sigma_1\delta_1\mu_1\Delta(f^E)_{i,j}^n \ - \ \tfrac{1}{2}\sigma_1\delta_1\left((\rho_1^E)^n\delta_1\Delta w^n\right)_{i,j} \\ + \ \sigma_2\delta_2\mu_2\Delta(g^E)_{i,j}^n \ - \ \tfrac{1}{2}\sigma_2\delta_2\left((\rho_2^E)^n\delta_2\Delta w^n\right)_{i,j} \ = \ -\Delta t\cdot\mathcal{R}_{i,j}^n\end{aligned}$$

The matrix-free treatment can thus be seen as a non-consistent (in the general case) implicitation of a given upwind explicit stage - the implicit and explicit stages being consistent when the Rusanov scheme is also used for the explicit space discretization, which is rarely done in practice because of the large numerical dissipation associated with this scheme. The idea of coupling / combining various implicit and explicit stages in order to improve the global efficiency properties of a flow solver has been investigated for instance in [1], but in the context of block-implicit schemes. The recent work [51], where the above Rusanov-type implicit stage is now coupled with an HLL-based explicit stage after having been used in conjunction with other upwind explicit stages, illustrates the flexibility of such a non-consistent choice. Naturally, the global efficiency of the matrix-free treatment depends on the unit cost that can be achieved when actually solving the implicit stage (4.2.7)-(4.2.8) : this point is detailed hereafter, in particular in the case of all-speed flows, which introduces a preconditioning matrix in the implicit stage.

4.2.2 Resolution of the Matrix-Free scheme

In this paragraph, we want to outline how to solve the Matrix-Free implicit stage which contains variable increment vector Δw^n and flux increment vectors $\Delta(f^E)^n$ and $\Delta(g^E)^n$. To achieve this task, an iterative technique is used and the flux increment vectors are first relaxed as follows :

$$\begin{aligned}C_0\Delta w_{i,j}^{(l+1)} \ + \ C_1^-\Delta w_{i-1,j}^{(l+1)} + \ C_1^+\Delta w_{i+1,j}^{(l+1)} + \ C_2^-\Delta w_{i,j-1}^{(l+1)} + \ C_2^+\Delta w_{i,j+1}^{(l+1)} \\ = \ -\Delta t\cdot\mathcal{R}_{i,j}^n \ - \ \sigma_1\delta_1\mu_1\Delta(f^E)_{i,j}^{(l)} \ - \ \sigma_2\delta_2\mu_2\Delta(g^E)_{i,j}^{(l)}\end{aligned}$$

where the exponent (l) denotes the counter of the iterative method. In order to reduce the memory storage of the method, one can employ the Point Jacobi relaxation procedure which is especially easy to implement on unstructured grids. All extra-diagonal terms are also relaxed, thus, the Matrix-Free Point Jacobi (MF-PJ) scheme reads :

$$\begin{aligned}\Delta w_{i,j}^{(l+1)} \;=\; \frac{1}{C_0}\cdot\Big[\; -\Delta t\cdot\mathcal{R}_{i,j}^{n} \;-\; \sigma_1\delta_1\mu_1\Delta(f^E)_{i,j}^{(l)} \;-\; \sigma_2\delta_2\mu_2\Delta(g^E)_{i,j}^{(l)} \\ + C_1^-\Delta w_{i-1,j}^{(l)} + \; C_1^+\Delta w_{i+1,j}^{(l)} + \; C_2^-\Delta w_{i,j-1}^{(l)} + \; C_2^+\Delta w_{i,j+1}^{(l)} \;\Big]\end{aligned} \tag{4.2.11}$$

This Matrix-Free formulation does not require any matrix inversion since the coefficient C_0 is purely scalar. The method is therefore very cheap in terms of both cost per iteration and memory storage. Of course, the Point Jacobi procedure is well known to display a poor intrinsic efficiency : a thorough analysis is therefore required to determine whether the low-cost of the treatment can nonetheless balance this poor efficiency; such a study, relying on Von Neumann analysis for evaluating intrinsic efficiency and on test-problems in order to take into account the actual cost of the approach, will be performed in the following chapters.
The Matrix-Free implicit scheme could be improved, as far as intrinsic efficiency is concerned, by resorting to other solution methods, such as the line-treatments described in the previous chapters. However, since we also require the method to remain easy to implement on unstructured grids, we rather turn toward the Symmetric Gauss-Seidel relaxation procedure, which seems a good candidate to ensure such requirements, that is to improve over Point Jacobi concerning intrinsic efficiency while remaining easy to implement on unstructured grids. Indeed, we previously showed that this procedure enables the implicit Block scheme to recover the Direct Solver efficiency with a number of inner-iterations much reduced with respect to Point Jacobi. The next chapters, including Von Neumann analysis as well as validation test-cases, will be dedicated to the study of this procedure coupled with the Matrix-Free implicit stage that has just been described. Meanwhile, let us describe how to modify the PJ algorithm so as to adapt it to the SGS strategy. The MF-PJ algorithm reads :

$$\left\{\begin{array}{l}\Delta w_{i,j}^{(0)} = 0\ , \quad l=0,\ p \\ \Delta w_{i,j}^{(l+1)} = \dfrac{1}{C_0}\cdot\Big[-\Delta t\cdot\mathcal{R}_{i,j}^{n} + \sigma_1\dfrac{1}{2}\Delta(f^E)_{i-1,j}^{(l)} + C_1^-\Delta w_{i-1,j}^{(l)} + \sigma_2\dfrac{1}{2}\Delta(g^E)_{i,j-1}^{(l)} + C_2^-\Delta w_{i,j-1}^{(l)} \\ \qquad\qquad -\sigma_1\dfrac{1}{2}\Delta(f^E)_{i+1,j}^{(l)} + C_1^+\Delta w_{i+1,j}^{(l)} - \sigma_2\dfrac{1}{2}\Delta(g^E)_{i,j+1}^{(l)} + C_2^+\Delta w_{i,j+1}^{(l)}\Big] \\ w_{i,j}^{n+1} = w_{i,j}^{n} + \Delta w_{i,j}^{(p)}\end{array}\right. \tag{4.2.12}$$

The SGS algorithm makes immediately use of the states that are being evaluated during the forward sweep (resp. the backward sweep) in order to compute the next ones. Thus, the new

algorithm reads :

$$
\left\{
\begin{array}{l}
\Delta w_{i,j}^{(0)} = 0 \ , \quad l = 0, \ p \\
\\
\text{Forward sweep :} \\
\\
\Delta w_{i,j}^{(*)} = \dfrac{1}{C_0} \cdot \Big[-\Delta t \cdot \mathcal{R}_{i,j}^{n} + \sigma_1 \dfrac{1}{2} \Delta (f^E)_{i-1,j}^{(*)} + C_1^- \Delta w_{i-1,j}^{(*)} + \sigma_2 \dfrac{1}{2} \Delta (g^E)_{i,j-1}^{(*)} + C_2^- \Delta w_{i,j-1}^{(*)} \\
\qquad\qquad\qquad\qquad - \sigma_1 \dfrac{1}{2} \Delta (f^E)_{i+1,j}^{(l)} + C_1^+ \Delta w_{i+1,j}^{(l)} - \sigma_2 \dfrac{1}{2} \Delta (g^E)_{i,j+1}^{(l)} + C_2^+ \Delta w_{i,j+1}^{(l)} \Big] \\
\\
\text{Backward sweep :} \\
\\
\Delta w_{i,j}^{(l+1)} = \dfrac{1}{C_0} \cdot \Big[-\Delta t \cdot \mathcal{R}_{i,j}^{n} + \sigma_1 \dfrac{1}{2} \Delta (f^E)_{i-1,j}^{(*)} + C_1^- \Delta w_{i-1,j}^{(*)} + \sigma_2 \dfrac{1}{2} \Delta (g^E)_{i,j-1}^{(*)} + C_2^- \Delta w_{i,j-1}^{(*)} \\
\qquad\qquad\qquad\qquad - \sigma_1 \dfrac{1}{2} \Delta (f^E)_{i+1,j}^{(l+1)} + C_1^+ \Delta w_{i+1,j}^{(l+1)} - \sigma_2 \dfrac{1}{2} \Delta (g^E)_{i,j+1}^{(l+1)} + C_2^+ \Delta w_{i,j+1}^{(l+1)} \Big] \\
w_{i,j}^{n+1} = w_{i,j}^{n} + \Delta w_{i,j}^{(p)}
\end{array}
\right. \tag{4.2.13}
$$

It is clear that the above algorithm does not add complexity to the method since it only requires the inversion of the coefficient C_0. Moreover it still does not involve any matrix storage. In the rest of the thesis, it will be denoted MF-SGS. As explained earlier, the MF-SGS scheme is expected to provide a better efficiency than the MF-PJ scheme. We will study the properties of theses schemes in the next chapters when successively applied to inviscid, viscous and unsteady flow problems. Therefore, let us describe in the next section the extension of the Matrix-Free framework to viscous flows.

4.2.3 Extension to viscous flows

In the first chapter, we retained a simplified implicit stage for the viscous fluxes in order to preserve the diagonal dominance of the I/III system. The numerical viscous fluxes at time level $(n+1)$ are therefore linearized as follows :

$$
(F^V)_{i+1/2,j}^{n+1} \approx (F^V)_{i+1/2,j}^{n} + (A_1^V)_{i+1/2,j}^{n} \frac{(\delta_1 \Delta w)_{i+1/2,j}^{n}}{\delta x}
$$

$$
(G^V)_{i,j-1/2}^{n+1} \approx (G^V)_{i,j-1/2}^{n} + (B_2^V)_{i,j-1/2}^{n} \frac{(\delta_2 \Delta w)_{i,j-1/2}^{n}}{\delta y}
$$

where we recall that $A_1^V = \dfrac{\partial f^V}{\partial w_x}$ and $B_2^V = \dfrac{\partial g^V}{\partial w_y}$. So as to preserve the Matrix-Free property, matrices A_1^V and B_2^V are replaced by their viscous spectral radius. In appendix A, it is shown for the sake of completeness that both matrices possess the same spectral radius :

$$
\rho(A_1^V) \ = \ \rho(B_2^V) \ = \ \rho^V \ = \ \frac{\gamma \, \mu}{Pr \ Re \ \rho} \tag{4.2.14}
$$

However, in the rest of the thesis, for the sake of clarity, we will distinguish the viscous spectral radius in each direction by setting :

$$\rho_1^V \;=\; \rho\left(\frac{A_1^V}{\delta x}\right) \qquad \text{and} \qquad \rho_2^V \;=\; \rho\left(\frac{B_2^V}{\delta y}\right) \tag{4.2.15}$$

Hence, the simplified numerical fluxes read :

$$(F^V)^{n+1}_{i+1/2,j} \approx (F^V)^n_{i+1/2,j} + (\rho_1^V)^n_{i+1/2,j}(\delta_1 \Delta w)^n_{i+1/2,j}$$

$$(G^V)^{n+1}_{i,j-1/2} \approx (G^V)^n_{i,j-1/2} + (\rho_2^V)^n_{i,j-1/2}(\delta_2 \Delta w)^n_{i,j-1/2}$$

Inserting these expressions into the implicit stage yields :

$$C_0\Delta w^n_{i,j} \;+\; C_1^-\Delta w^n_{i-1,j} + \; C_1^+\Delta w^n_{i+1,j} + \; C_2^-\Delta w^n_{i,j-1} + \; C_2^+\Delta w^n_{i,j+1}$$
$$= \; -\Delta t \cdot \mathcal{R}^n_{i,j} \;-\; \sigma_1\delta_1\mu_1\Delta(f^E)^n_{i,j} \;-\; \sigma_2\delta_2\mu_2\Delta(g^E)^n_{i,j}$$

where the coefficients C take into account the viscous contribution :

$$\begin{cases} C_1^- = -\sigma_1\left(\frac{1}{2}\rho_1^E + \rho_1^V\right)^n_{i-\frac{1}{2},j} \\ C_1^+ = -\sigma_1\left(\frac{1}{2}\rho_1^E + \rho_1^V\right)^n_{i+\frac{1}{2},j} \\ C_2^- = -\sigma_2\left(\frac{1}{2}\rho_2^E + \rho_2^V\right)^n_{i,j-\frac{1}{2}} \\ C_2^+ = -\sigma_2\left(\frac{1}{2}\rho_2^E + \rho_2^V\right)^n_{i,j+\frac{1}{2}} \\ C_0 \;= 1 - C_1^- - C_1^+ - C_2^- - C_2^+ \end{cases} \tag{4.2.16}$$

Besides, the explicit stage reads :

$$\mathcal{R}^n_{i,j} = \left(\frac{\delta_1(F^E - F^V)}{\delta x} + \frac{\delta_2(G^E - G^V)}{\delta y}\right)^n_{i,j} .$$

Hence, the Matrix-Free implicit treatment extends straightforwardly to viscous flows. Indeed, thanks to the viscous spectral radius simplification, there is no need to store the viscous Jacobian matrices and the simplicity of the method is preserved. The scheme can then be solved using SGS procedure or PJ method as well. In chapter 6, the impact of the viscous spectral radius simplification on the efficiency of the scheme will be studied. In the next paragraph, the extension of the method to the resolution of unsteady flows is presented.

4.2.4 Extension to unsteady flows

In the first chapter, the dual-time stepping formulation was introduced so as to compute unsteady flows. We showed that this method enables us to use the convergence speedup strategies ordinarily applied for steady flow calculations. In this paragraph, we will outline how the dual-time stepping scheme can be also easily adapted to the Matrix-Free framework.
Let us first recall the expression of the dual-time stepping discretization :

$$\frac{w^{n,m+1}_{i,j} - w^n_{i,j}}{\Delta\tau} + \frac{3w^{n,m+1}_{i,j} - 4w^n_{i,j} + w^{n-1}_{i,j}}{2\Delta t} + \left(\frac{\delta_1(F^E - F^V)}{\delta x} + \frac{\delta_2(G^E - G^V)}{\delta y}\right)^{n,m+1}_{i,j} = 0$$

where $\Delta\tau$ is the pseudo-time step and $w^{n,m+1}$ denotes the state at pseudo-time $(m+1)\Delta\tau$ in the convergence process toward the physical state at time $(n+1)\Delta t$. The Matrix-Free simplifications can be simply applied during the linearization of the numerical fluxes around the pseudo-time level (n,m). As far as the inviscid numerical fluxes are concerned, we have :

$$\begin{aligned}(F^E)^{n,m+1} &\approx (F^E)^{n,m} + \mu_1\Delta(f^E)^{n,m} - \tfrac{1}{2}(\rho_1^E)^{n,m}\delta_1\Delta w^{n,m} \ , \\ (G^E)^{n,m+1} &\approx (G^E)^{n,m} + \mu_2\Delta(g^E)^{n,m} - \tfrac{1}{2}(\rho_2^E)^{n,m}\delta_2\Delta w^{n,m} \ .\end{aligned} \tag{4.2.17}$$

Similarly, the viscous spectral radius simplification yields :

$$\begin{aligned}(F^V)^{n,m+1} &\approx (F^V)^{n,m} + (\rho_1^V)^{n,m}\delta_1\Delta w^{n,m} \ , \\ (G^V)^{n,m+1} &\approx (G^V)^{n,m} + (\rho_2^V)^{n,m}\delta_2\Delta w^{n,m} \ .\end{aligned} \tag{4.2.18}$$

Inserting these expression into the implicit stage yields :

$$\begin{aligned}C_0\Delta w_{i,j}^{n,m} \ + \ C_1^-\Delta w_{i-1,j}^{n,m} + \ C_1^+\Delta w_{i+1,j}^{n,m} + \ C_2^-\Delta w_{i,j-1}^{n,m} + \ C_2^+\Delta w_{i,j+1}^{n,m} \\ = \ -\Delta\tau\cdot\mathcal{R}_{i,j}^{n,m} \ - \ \sigma_1\delta_1\mu_1\Delta(f^E)_{i,j}^{n,m} \ - \ \sigma_2\delta_2\mu_2\Delta(g^E)_{i,j}^{n,m}\end{aligned} \tag{4.2.19}$$

where the coefficients C are now given by :

$$\begin{cases} C_1^- = -\sigma_1\left(\frac{1}{2}\rho_1^E + \rho_1^V\right)_{i-\frac{1}{2},j}^{n,m} \\ C_1^+ = -\sigma_1\left(\frac{1}{2}\rho_1^E + \rho_1^V\right)_{i+\frac{1}{2},j}^{n,m} \\ C_2^- = -\sigma_2\left(\frac{1}{2}\rho_2^E + \rho_2^V\right)_{i,j-\frac{1}{2}}^{n,m} \\ C_2^+ = -\sigma_2\left(\frac{1}{2}\rho_2^E + \rho_2^V\right)_{i,j+\frac{1}{2}}^{n,m} \\ C_0 \ = (1+\frac{3}{2}\lambda) - C_1^- - C_1^+ - C_2^- - C_2^+ \end{cases} \tag{4.2.20}$$

with

$$\lambda = \frac{\Delta\tau}{\Delta t} \ , \quad \sigma_1 = \frac{\Delta\tau}{\delta x} \ , \quad \sigma_2 = \frac{\Delta\tau}{\delta y} \ .$$

The operator $\mathcal{R}_{i,j}^{n,m}$ remains unchanged with respect to the block-implicit scheme :

$$\mathcal{R}_{i,j}^{n,m} = \frac{3w_{i,j}^{n,m} - 4w_{i,j}^{n} + w_{i,j}^{n-1}}{2\Delta t} + \left(\frac{\delta_1(F^E - F^V)}{\delta x} + \frac{\delta_2(G^E - G^V)}{\delta y}\right)_{i,j}^{n,m}$$

Eventually, SGS relaxation procedure as well as PJ algorithm can be employed to solve the Matrix-Free dual-time stepping scheme. Thus, the extension of the Matrix-Free implicit treatment to unsteady flows does not add complexity. We now want to study the impact of the low-Mach preconditioning on the MF formulation.

4.2.5 Impact of the low-Mach preconditioning

In the first chapter, we showed that the low-Mach preconditioning only modifies the pseudo-time derivative term and the dissipation of the numerical inviscid fluxes. The preconditioning has therefore an impact on the spectral radius simplification, indeed, the preconditioned Rusanov scheme reads :

$$\tilde{F}^{Rus}_{i+\frac{1}{2},j} = (\mu_1(f^E) - \tfrac{1}{2}\rho(PA^E)P^{-1}\cdot\delta_1 w)_{i+\frac{1}{2},j} = (\mu_1(f^E) - \tfrac{1}{2}\tilde{\rho}^E_1 P^{-1}\cdot\delta_1 w)_{i+\frac{1}{2},j}$$

$$\tilde{G}^{Rus}_{i,j-\frac{1}{2}} = (\mu_2(g^E) - \tfrac{1}{2}\rho(PB^E)P^{-1}\cdot\delta_2 w)_{i,j-\frac{1}{2}} = (\mu_2(g^E) - \tfrac{1}{2}\tilde{\rho}^E_2 P^{-1}\cdot\delta_2 w)_{i,j-\frac{1}{2}}$$

As a result, applying the substitution of the numerical dissipation matrices yields in this preconditioned case :

$$\begin{aligned}
(\tilde{F^E})^{n,m+1} &\approx (\tilde{F^E})^{n,m} + \mu_1(\Delta f^E)^{n,m} - \tfrac{1}{2}\tilde{\rho}^E_1 P^{-1}\cdot\delta_1\Delta w^{n,m} \ , \\
(\tilde{G^E})^{n,m+1} &\approx (\tilde{G^E})^{n,m} + \mu_2(\Delta g^E)^{n,m} - \tfrac{1}{2}\tilde{\rho}^E_2 P^{-1}\cdot\delta_2\Delta w^{n,m} \ ,
\end{aligned} \tag{4.2.21}$$

and inserting this linearization into the implicit stage leads to :

$$\begin{aligned}
C_0\Delta w^{n,m}_{i,j} \ + \ C_1^-\Delta w^{n,m}_{i-1,j} + \ C_1^+\Delta w^{n,m}_{i+1,j} + \ C_2^-\Delta w^{n,m}_{i,j-1} + \ C_2^+\Delta w^{n,m}_{i,j+1} & \\
= \ -\Delta\tau\cdot\mathcal{R}^{n,m}_{i,j} \ - \ \sigma_1\delta_1\mu_1\Delta(f^E)^{n,m}_{i,j} \ - \ \sigma_2\delta_2\mu_2\Delta(g^E)^{n,m}_{i,j} &
\end{aligned} \tag{4.2.22}$$

with

$$\mathcal{R}^{n,m}_{i,j} = \frac{3w^{n,m}_{i,j} - 4w^n_{i,j} + w^{n-1}_{i,j}}{2\Delta t} + \left(\frac{\delta_1(\tilde{F}^E - F^V)}{\delta x} + \frac{\delta_2(\tilde{G}^E - G^V)}{\delta y}\right)^{n,m}_{i,j}$$

and

$$\begin{cases}
C_1^- = -\sigma_1\left(\tfrac{1}{2}\tilde{\rho}^E_1 P^{-1} + \rho^V_1 Id\right)^{n,m}_{i-\frac{1}{2},j} \\
C_1^+ = -\sigma_1\left(\tfrac{1}{2}\tilde{\rho}^E_1 P^{-1} + \rho^V_1 Id\right)^{n,m}_{i+\frac{1}{2},j} \\
C_2^- = -\sigma_2\left(\tfrac{1}{2}\tilde{\rho}^E_2 P^{-1} + \rho^V_2 Id\right)^{n,m}_{i,j-\frac{1}{2}} \\
C_2^+ = -\sigma_2\left(\tfrac{1}{2}\tilde{\rho}^E_2 P^{-1} + \rho^V_2 Id\right)^{n,m}_{i,j+\frac{1}{2}} \\
C_0 \ = P^{-1} + \tfrac{3}{2}\lambda Id - C_1^- - C_1^+ - C_2^- - C_2^+
\end{cases} \tag{4.2.23}$$

Thus, the diagonal coefficient C_0 given by (4.2.23) is now a full matrix, to be inverted, and coefficient $C^{\pm}$ are also block coefficients, so that storage requirement and operations count of the implicit treatment increase for low-Mach number flows. The next section explains how it is possible to take advantage of the properties of the preconditioning matrix P to keep the preconditioned approach as simple as its compressible flow version.

4.3 Preserving the Matrix-Free property with Preconditioning

4.3.1 General framework

Before detailing the Matrix-Free method for low-Mach number flows, let us introduce a general framework to compute the time-accurate Navier-Stokes equations on multi-dimensional (2D or 3D) Cartesian grids : this gives us the opportunity to show the numerical developments that are almost constantly presented in 2D in this thesis for the sake of simplicity can be extended to the 3D case in a genuinely straightforward fashion. The multidimensional governing equations read :

$$P^{-1}\frac{\partial w}{\partial \tau}+\frac{\partial w}{\partial t}+\sum_{p=1}^{d}\frac{\partial f_p^E}{\partial x_p}=\sum_{p=1}^{d}\frac{\partial f_p^V}{\partial x_p}, \tag{4.3.1}$$

where we recall that w is the vector of the conserved variable, τ is the pseudo-time, t is the physical time, f_p^E and f_p^V are respectively the convective and viscous fluxes in the p^{th} space-direction (d is the dimension of the problem) and P is a preconditioning matrix which takes, in the present work, the form proposed in [75] as described in details in section 1.3.2 - the properties of this preconditioning matrix will be further recalled in section 4.3.2. To solve system (4.3.1), the following numerical scheme is used :

$$\left(P_c^{-1}\right)_j^{n,m}\frac{\Delta w_j^{n,m}}{\Delta \tau_j^{n,m}}+\frac{\frac{3}{2}(w_j^{n,m+1}-w_j^n)-\frac{1}{2}\Delta w_j^{n-1}}{\Delta t}+\sum_{p=1}^{d}\left(\frac{\delta_p h_p}{\delta x_p}\right)_j^{n,m+1}=0, \tag{4.3.2}$$

where m is the pseudo-iteration counter, n is the time step counter, $\Delta w^{n,m}=w^{n,m+1}-w^{n,m}$, $\Delta w^{n-1}=w^n-w^{n-1}$, $j=(j_1,j_2,\ldots,j_d)$ is a multi-integer associated with a point $x_j=(j_1\delta x_1,\ldots,j_d\delta x_d)$ of a regular Cartesian grid. $(\delta_p h_p)$ is the difference operator on a grid cell δ_p in the p^{th} space-direction applied to the numerical flux h_p approximating the physical flux $(f_p^E-f_p^V)$:

$$(\delta_p h_p)_j=(h_p)_{j+\frac{e_p}{2}}-(h_p)_{j-\frac{e_p}{2}}$$

where e_p is a multi-integer with the q-th component e_{pq} which is equal to 0 if $q\neq p$ and to 1 if $q=p$. Equation (4.3.2) can be written in the following form :

$$\left(P_c^{-1}\right)_j^{n,m}\frac{\Delta w_j^{n,m}}{\Delta \tau_j^{n,m}}+\frac{3}{2}\frac{\Delta w_j^{n,m}}{\Delta t}+\sum_{p=1}^{d}\left(\frac{\delta_p(\Delta h_p)}{\delta x_p}\right)_j^{n,m}=-\left(\mathcal{R}\right)_j^{n,m}, \tag{4.3.3}$$

with an explicit stage given by :

$$\left(\mathcal{R}\right)_j^{n,m}=\frac{\frac{3}{2}(w_j^{n,m}-w_j^n)-\frac{1}{2}\Delta w_j^{n-1}}{\Delta t}+\sum_{p=1}^{d}\left(\frac{\delta_p h_p}{\delta x_p}\right)_j^{n,m}.$$

When the steady state with respect to pseudo-time τ is reached, the LHS of (4.3.3) goes to zero, hence, the numerical approach used to compute the LHS of equation (4.3.3) does not affect the (space and time) accuracy of the method (provided that the convergence on m is reached). We can then apply the Matrix-Free simplifications on the LHS terms. On the other hand, in the RHS of (4.3.3), the inviscid fluxes are computed using classical inviscid numerical fluxes (e.g. Roe, Rusanov AUSM$^+$...), extended to higher order using a variable reconstruction approach

(MUSCL) and with a numerical dissipation taking into account the preconditioning of (4.3.1). Viscous fluxes are approximated at second-order using centered formula (*cf.* chapter 1).

In order to apply the Matrix-Free method that was described in the previous sections, the inviscid fluxes of the LHS are evaluated using the first-order accurate preconditioned Rusanov scheme :

$$h_p^E = \mu_p f_p^E - \frac{1}{2} P^{-1} \rho(P A_p^E) \delta_p w \ , \tag{4.3.4}$$

where μ_p is the average operator over a grid cell in the p^{th} space direction :

$$(\mu_p \phi_p)_j = \frac{1}{2}\left((\phi_p)_{j+\frac{e_p}{2}} + (\phi_p)_{j-\frac{e_p}{2}} \right) \ ,$$

and $\rho(PA_p^E) = \tilde{\rho}_p^E$ denotes the spectral radius of the preconditioned Jacobian matrix $(P \cdot \dfrac{\partial f_p^E}{\partial w})$. Introducing formula (4.3.4) into the expression of $\left(\Delta(h_p^E)\right)$ leads to :

$$\begin{aligned} \left(\Delta h_p^E\right)_{j\pm e_p/2}^{n,m} &= \mu_p \left(\Delta {f_p}^E\right)_{j\pm e_p}^{n,m} \pm \frac{1}{2} \Delta \left(\left(P^{-1} \tilde{\rho}_p^E\right)_{j\pm e_p/2} \left(w_j - w_{j\pm e_p}\right) \right)^{n,m} \\ &\approx \mu_p \left(\Delta {f_p}^E\right)_{j\pm e_p}^{n,m} \pm \frac{1}{2} \left(P^{-1} \tilde{\rho}_p^E\right)_{j\pm e_p/2}^{n,m} \Delta \left(w_j - w_{j\pm e_p}\right)^{n,m} \end{aligned}$$

It follows that :

$$\begin{aligned} \left(\delta_p \left(\Delta h_p^E\right)\right)_j^{n,m} = \delta_p \mu_p \left(\Delta {f_p}^E\right)_{j+e_p}^{n,m} &+ \frac{1}{2} \left(\left(P^{-1} \tilde{\rho}_p^E\right)_{j+e_p/2}^{n,m} + \left(P^{-1} \tilde{\rho}_p^E\right)_{j-e_p/2}^{n,m} \right) \Delta w_j^{n,m} \\ &- \frac{1}{2} \left(P^{-1} \tilde{\rho}_p^E\right)_{j+e_p/2}^{n,m} \Delta w_{j+e_p}^{n,m} - \frac{1}{2} \left(P^{-1} \tilde{\rho}_p^E\right)_{j-e_p/2}^{n,m} \Delta w_{j-e_p}^{n,m} \end{aligned} \tag{4.3.5}$$

As far as the viscous fluxes of the LHS are concerned, we neglect their dependence with respect to the tangential derivatives :

$$\left(h_p^V\right)_{j\pm e_p/2} = \left(A_p^V \frac{\delta w}{\delta x_p} \right)_{j\pm e_p/2} = \left(Q_p^V\right)_{j\pm e_p/2} \left(\delta_p w\right)_{j\pm e_p/2} \ ,$$

then, we proceed to the viscous spectral radius simplification as follows :

$$\left(h_p^V\right)_{j\pm e_p/2} \approx \rho(Q_p^V)_{j\pm e_p/2} \left(\delta_p w\right)_{j\pm e_p/2} = \left(\rho_p^V\right)_{j\pm e_p/2} \left(\delta_p w\right)_{j\pm e_p/2} \ , \tag{4.3.6}$$

so that :

$$\begin{aligned} \left(\delta_p (\Delta h_p^V)\right)_j^{n,m} = &- \left((\rho_p^V)_{j+e_p/2}^{n,m} + (\rho_p^V)_{j-e_p/2}^{n,m} \right) \Delta w_j^{n,m} \\ &+ (\rho_p^V)_{j+e_p/2}^{n,m} \Delta w_{j+e_p}^{n,m} + (\rho_p^V)_{j-e_p/2}^{n,m} \Delta w_{j-e_p}^{n,m} \end{aligned} \tag{4.3.7}$$

Inserting (4.3.5) and (4.3.7) into (4.3.3) yields the following simple implicit scheme :

$$D_j^{n,m}\Delta w_j^{n,m} + \sum_p \left\{ \left(\frac{\delta_p \mu_p}{\delta x_p} \Delta (f_p^E)^{n,m} \right)_j - \left(C_p^-\right)_j^{n,m} \Delta w_{j-e_p}^{n,m} - \left(C_p^+\right)_j^{n,m} \Delta w_{j+e_p}^{n,m} \right\} = -\left(\mathcal{R}\right)_j^{n,m}, \tag{4.3.8}$$

where

$$D_j^{n,m} = \left(\frac{1}{\Delta\tau} P_c^{-1} \right)_j^{n,m} + \frac{3}{2\Delta t} I_d + \sum_p \left\{ \left(C_p^+\right)_j^{n,m} + \left(C_p^-\right)_j^{n,m} \right\}, \tag{4.3.9}$$

$$\left(C_p^{\pm}\right)_j^{n,m} = \frac{1}{\delta x_p} \left(\frac{1}{2} \tilde{\rho}_p^E P^{-1} + \tilde{\rho}_p^V \right)_{j \pm e_p/2}^{n,m}. \tag{4.3.10}$$

This implicit scheme is solved using a relaxation procedure such as Point Jacobi (or Symmetric Gauss-Seidel) which leads to :

$$D_j^{n,m}\Delta w_j^{(l+1)} = -\left(\mathcal{R}\right)_j^{n,m} - \sum_p \left\{ \left(\frac{\delta_p \mu_p}{\delta x_p} \Delta (f_p^E)^{(l)} \right)_j - \left(C_p^-\right)_j^{n,m} \Delta w_{j-e_p}^{(l)} - \left(C_p^+\right)_j^{n,m} \Delta w_{j+e_p}^{(l)} \right\} \tag{4.3.11}$$

Hence, the coefficient D has to be inverted in order to obtain the increment $\Delta w^{(l+1)}$. As mentioned in the previous paragraph, the inversion is achieved directly when compressible flows are computed since, in this case, the matrix D is purely scalar. However, when low-Mach number flows are computed, the preconditioning is turned on so that P is no longer the identity matrix and consequently D becomes a full matrix as well as the coefficients $C^{\pm}$. Nevertheless, it is possible to take advantage of the properties of the preconditioning matrix P to keep the preconditioned approach as simple as its compressible version.

4.3.2 Detail of the preconditioning matrix

We showed in the first chapter that the Turkel preconditioner could be written under the following form :

$$P = I_d + (\beta^2 - 1) \cdot Q$$

with

$$Q = \frac{\gamma - 1}{c^2} \begin{bmatrix} q^2 = \frac{1}{2}(u^2+v^2) & -u & -v & 1 \\ uq^2 & -u^2 & -uv & u \\ vq^2 & -uv & -v^2 & v \\ Hq^2 & -uH & -vH & H \end{bmatrix} = \frac{\gamma - 1}{c^2} \left[\begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix} \cdot \left(q^2 \; ; \; -u \; ; \; -v \; ; \; 1 \right) \right]$$

The matrix Q was shown to be idempotent, *i.e.* $Q^2 = Q$. This property is very useful since it enables us to invert in a simple way the preconditioning matrix P :

$$P^{-1} = I_d + \frac{1}{\beta^2 - 1} Q$$

Now, if the matrix $D_j^{n,m}$ is calculated using a single evaluation of the matrix P^{-1} at point j, it can be written as follows :

$$D_j^{n,m} = a_j^{n,m} \cdot \left(P^{-1}\right)_j^{n,m} + b_j^{n,m} I_d$$

with

$$\begin{cases} a_j^{n,m} = \dfrac{1}{\Delta\tau_j^{n,m}} + \dfrac{1}{2}\displaystyle\sum_{p=1}^{d} \dfrac{1}{\delta x_p} \left\{ \left(\tilde{\rho}_p^E\right)_{j+e_p/2}^{n,m} + \left(\tilde{\rho}_p^E\right)_{j-e_p/2}^{n,m} \right\} \\ b_j^{n,m} = \dfrac{3}{2\Delta t} + \displaystyle\sum_{p=1}^{d} \dfrac{1}{\delta x_p} \left\{ \left(\rho_p^V\right)_{j+e_p/2}^{n,m} + \left(\rho_p^V\right)_{j-e_p/2}^{n,m} \right\} \end{cases}$$

Then using the property of Q, the invert of the matrix $D_j^{n,m}$ is computed explicitly as follows :

$$\left(D^{-1}\right)_j^{n,m} = \left(\frac{1}{a+b} \left(I_d + \frac{a\left(\beta^2 - 1\right)}{a + b\beta^2} Q_c \right) \right)_j^{n,m} \tag{4.3.12}$$

Besides, if in the coefficients $C^{\pm}$, the matrices P^{-1} are evaluated at the point j, it is possible to rewrite the scheme (4.3.11) in the form :

$$\Delta w_j^{(l+1)} = \left(D^{-1}\right)_j^{n,m} \cdot \left(\Delta w_1^{(l)}\right)_j + \left(D^{-1}\right)_j^{n,m} \cdot \left(P^{-1}\right)_j^{n,m} \cdot \left(\Delta w_2^{(l)}\right)_j$$

with the following definitions for the increments $\left(\Delta w_1^{(l)}\right)_j$ and $\left(\Delta w_2^{(l)}\right)_j$:

$$\begin{cases} \left(\Delta w_1^{(l)}\right)_j = -\left(\mathcal{R}\right)_j^{n,m} - \displaystyle\sum_p \dfrac{1}{\delta x_p} \Big\{ \Delta(f_p^E)_{j+e_p}^{(l)} - \left(\rho_p^V\right)_{j+e_p/2}^{n,m} \cdot \Delta w_{j+e_p}^{(l)} \\ \qquad\qquad\qquad\qquad - \Delta(f_p^E)_{j-e_p}^{(l)} - \left(\rho_p^V\right)_{j-e_p/2}^{n,m} \cdot \Delta w_{j-e_p}^{(l)} \Big\} \\ \left(\Delta w_2^{(l)}\right)_j = \displaystyle\sum_p \dfrac{1}{\delta x_p} \left\{ \left(\tilde{\rho}_p^E\right)_{j+e_p/2}^{n,m} \cdot \Delta w_{j+e_p}^{(l)} + \left(\tilde{\rho}_p^E\right)_{j-e_p/2}^{n,m} \cdot \Delta w_{j-e_p}^{(l)} \right\} \end{cases}$$

The matrix product $\left(D^{-1} \cdot P^{-1}\right)$ that appears in the expression of the implicit scheme can be also calculated explicitly thanks to the properties of each matrix :

$$\left(D^{-1}\right)_j^{n,m} \cdot \left(P^{-1}\right)_j^{n,m} = \left(\frac{1}{a+b} \left(I_d + \frac{b\left(\beta^2 - 1\right)}{a + b\beta^2} Q_c \right) \right)_j^{n,m} \tag{4.3.13}$$

These expressions enable us to build a truly Matrix-Free implicit treatment for all speed flows. Such a scheme is described in the next paragraph.

4.3.3 A truly Matrix-Free treatment for flows at all speeds

Using expressions (4.3.12) and (4.3.13) to evaluate $\Delta w_j^{(l+1)}$ yields the following implicit treatment :

$$\begin{aligned} \Delta w_j^{(l+1)} = & \frac{1}{(a+b)_j^{n,m}} \cdot \left[\Delta w_1^{(l)} + \Delta w_2^{(l)} \right]_j \\ & + \frac{\left(\beta^2\right)_j^{n,m} - 1}{\left((a+b)\left(a+b\beta^2\right)\right)_j^{n,m}} \cdot Q_j^{n,m} \cdot \left[a^{n,m} \Delta w_1^{(l)} + b^{n,m} \Delta w_2^{(l)} \right]_j \end{aligned} \tag{4.3.14}$$

This implicit treatment enables to decouple nicely the standard Matrix-Free implicit treatment used for the compressible flow (that is for $\beta = 1$ so that the second term cancels) from the added treatment specific to low-Mach number flows (in this case $\beta \neq 1$). Besides, treatment (4.3.14) becomes wholly Matrix-Free if the specific expression of matrix Q is employed to compute a matrix-vector product such as $Q \cdot X$ under the form :

$$Q \cdot X = \frac{\gamma - 1}{c^2} \left(q^2 X^{(1)} - u X^{(2)} - v X^{(3)} + X^{(4)} \right) \cdot \begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix}$$

with $X^{(m)}$ the components of vector X. Eventually, $\Delta w_j^{(l+1)}$ can be computed at a low cost and with a reduced memory storage using the following expression :

$$\Delta w_j^{(l+1)} = \frac{1}{(a+b)_j^{n,m}} \left[\Delta w_1^{(l)} + \Delta w_2^{(l)} \right]_j + \Delta \phi_j^{(l)} \cdot \begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix}_j^{n,m} \tag{4.3.15}$$

where the scalar $\Delta \phi_j^{(l)}$ is given by :

$$\Delta \phi_j^{(l)} = \chi_j^{n,m} \left[q^2 \left(a \Delta w_1^{(1)} + b \Delta w_2^{(1)} \right) - u \left(a \Delta w_1^{(2)} + b \Delta w_2^{(2)} \right) \right.$$
$$\left. - v \left(a \Delta w_1^{(3)} + b \Delta w_2^{(3)} \right) + \left(a \Delta w_1^{(4)} + b \Delta w_2^{(4)} \right) \right]_j^{(l)}$$

and with

$$\chi_j^{n,m} = \left[\frac{\gamma - 1}{c^2} \cdot \frac{\beta^2 - 1}{(a+b)(a+b\beta^2)} \right]_j^{n,m} .$$

Formula (4.3.15) summarizes the all-speed flow low-cost matrix-free treatment investigated in the present work. Note this formula is specifically written for a Point-Jacobi treatment of the implicit stage. The extension to the SGS framework can be performed straightforwardly. Indeed, during the forward and the backward sweeps, the increments $\Delta w_1^{(l)}$ and $\Delta w_2^{(l)}$ have just to be computed using the states that have already been evaluated. For sake of clarity, let us express the new formulation of such increments in the case of the forward sweep only :

$$\begin{cases} \left(\Delta w_1^{(*,l)} \right)_j = - \left(\mathcal{R} \right)_j^{n,m} - \displaystyle\sum_p \frac{1}{\delta x_p} \left\{ \Delta (f_p^E)_{j+e_p}^{(l)} - \left(\rho_p^V \right)_{j+e_p/2}^{n,m} \cdot \Delta w_{j+e_p}^{(l)} \right. \\ \qquad\qquad \left. - \Delta (f_p^E)_{j-e_p}^{(*)} - \left(\rho_p^V \right)_{j-e_p/2}^{n,m} \cdot \Delta w_{j-e_p}^{(*)} \right\} \\ \left(\Delta w_2^{(*,l)} \right)_j = \displaystyle\sum_p \frac{1}{\delta x_p} \left\{ \left(\tilde{\rho}_p^E \right)_{j+e_p/2}^{n,m} \cdot \Delta w_{j+e_p}^{(l)} + \left(\tilde{\rho}_p^E \right)_{j-e_p/2}^{n,m} \cdot \Delta w_{j-e_p}^{(*)} \right\} \end{cases}$$

where the exponant $(*, l)$ emphasizes the fact that the increments rely on both iterative levels (l) and $(*)$ (see also SGS algorithm (4.2.13)).

4.4 Treatment of the boundaries of the computational domain

4.4.1 Explicit treatment

A simple way to deal with the boundaries of the computational domain is to set the increments at the boundaries equal to zero :

$$\begin{cases} \Delta w_B^{(l)} = w_B^{(l)} - w_B^n = 0 \\ \\ \Delta \left(f_p^E\right)_B^{(l)} = \left(f_p^E\right)_B^{(l)} - \left(f_p^E\right)_B^n = 0 \end{cases}$$

Hence, this explicit treatment does not add any calculation into the the Matrix-Free algorithm. However, such a choice will be shown to be inefficient in the case of high aspect ratio grids. It is therefore necessary to enhance the treatment of the boundaries.

4.4.2 Implicit treatment

In order to improve the Matrix-Free implicit method, we can implement the following treatment. First, let be q the vector of the well-suited variables for the computation of the boundary conditions. Then, the increments at the boundaries can be expressed in function of the interior increments :

$$\Delta q_B^{(l)} = \mathcal{A} \cdot \Delta q_I^{(l)} \ ,$$

where $\mathcal{A}$ denotes the matrix which defines the boundary conditions. Eventually, the increments of the conservative variables are computed as follows :

$$\begin{cases} w_B^{(l)} = w(q_B^{(l)}) \\ \\ \left(f_p^E\right)_B^{(l)} = f_p^E(w_B^{(l)}) \end{cases} \Longrightarrow \quad \begin{cases} \Delta w_B^{(l)} = w_B^{(l)} - w_B^n \\ \\ \Delta \left(f_p^E\right)_B^{(l)} = \left(f_p^E\right)_B^{(l)} - \left(f_p^E\right)_B^n \end{cases}$$

Thus, it is just necessary to evaluate the matrix $\mathcal{A}$ so as to compute these increments. Next paragraph will detail the evaluation of this matrix for specific cases.

4.4.3 Matrix of the boundary conditions

For each case detailed in the following, the computation variables are first precised.

Adiabatic wall

The computation variables are naturally the "viscous" ones, namely $q = (p,\ u,\ v,\ T)^T$. The boundary conditions are given by :

$$\frac{\partial p}{\partial n} = 0 \ ; \quad u = 0 \ ; \quad v = 0 \ ; \quad \frac{\partial T}{\partial n} = 0 \ .$$

Consequently, we have :

$$p_B = p_I \ ; \quad u_B = 0 \ ; \quad v_B = 0 \ ; \quad T_B = T_I \ .$$

or

$$\Delta p_B^{(l)} = \Delta p_I^{(l)} \; ; \quad \Delta u_B^{(l)} = 0 \; ; \quad \Delta v_B^{(l)} = 0 \; ; \quad \Delta T_B^{(l)} = \Delta T_I^{(l)} \; .$$

The matrix is then defined as follows :

$$\mathcal{A} = \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & 0 & \\ & & & 1 \end{bmatrix}$$

Isothermal wall

We keep the "viscous" variables and the boundary conditions read :

$$\frac{\partial p}{\partial n} = 0 \; ; \quad u = 0 \; ; \quad v = 0 \; ; \quad T = T_{\mathrm{W}} \; .$$

Hence we have :

$$\Delta p_B^{(l)} = \Delta p_I^{(l)} \; ; \quad \Delta u_B^{(l)} = 0 \; ; \quad \Delta v_B^{(l)} = 0 \; ; \quad \Delta T_B^{(l)} = 0 \; .$$

The matrix is then defined as follows :

$$\mathcal{A} = \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{bmatrix}$$

Slip conditions

In this case, we can choose the primitive variables $q = (\rho, \; u, \; v, \; p)^T$ to define the boundary conditions :

$$\frac{\partial \rho}{\partial n} = 0 \; ; \quad V_n = 0 \; ; \quad \frac{\partial V_t}{\partial n} = 0 \; ; \quad \frac{\partial p}{\partial n} = 0 \; .$$

where V_n and V_t denote the normal and the tangential velocities. It follows that :

$$\Delta \rho_B^{(l)} = \Delta \rho_I^{(l)} \; ; \quad \Delta u_B^{(l)} = n_y^2 \Delta u_I^{(l)} - n_x n_y \Delta v_I^{(l)} \; ; \quad \Delta v_B^{(l)} = n_x^2 \Delta v_I^{(l)} - n_x n_y \Delta u_I^{(l)} \; ; \quad \Delta p_B^{(l)} = \Delta p_I^{(l)} \; .$$

The matrix is then defined as follows :

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & n_y^2 & -n_x n_y & 0 \\ 0 & -n_x n_y & n_x^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Subsonic inlet

In this case, we impose the inlet enthalpy, the inlet entropy and the flow angle while we extrapolate the pressure from the interior of the domain :

$$H_B = H_\infty \; ; \quad S_B = S_\infty \; ; \quad \frac{(V_t)_B}{(V_n)_B} = \alpha_\infty \; ; \quad p_B = p_I \; .$$

The vector of the computation variables is therefore $q = (H,\ S,\ \alpha,\ p)^T$ and the matrix $\mathcal{A}$ reads :

$$\mathcal{A} = \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 1 \end{bmatrix}$$

Then, the primitive variables are computed as follows :

$$\begin{cases} \Delta p_B^{(l)} = \Delta p_I^{(l)} \\ \Delta \rho_B^{(l)} = \rho_B^{(l)} - \rho_B^n = \left(\dfrac{p_I^{(l)}}{S_\infty} \right)^{1/\gamma} - \left(\dfrac{p_I^n}{S_\infty} \right)^{1/\gamma} \\ (\Delta V_n)_B^{(l)} = \sqrt{\dfrac{2}{1+\alpha_\infty^2}} \cdot \left(\sqrt{H_\infty - \dfrac{\gamma}{\gamma-1} \dfrac{\left(p_I^{(l)}\right)^{(\gamma-1)/\gamma}}{S_\infty^{1/\gamma}}} - \sqrt{H_\infty - \dfrac{\gamma}{\gamma-1} \dfrac{\left(p_I^n\right)^{(\gamma-1)/\gamma}}{S_\infty^{1/\gamma}}} \right) \\ (\Delta V_t)_B^{(l)} = \alpha_\infty \cdot (\Delta V_n)_B^{(l)} \end{cases}$$

Incompressible inlet

The vector of the primitive variable is the best-suited for such conditions since we have :

$$\rho_B = \rho_\infty \; ; \quad u_B = u_\infty \; ; \quad v_B = v_\infty \; ; \quad p_B = p_I \; .$$

or

$$\Delta \rho_B^{(l)} = 0 \; ; \quad \Delta u_B^{(l)} = 0 \; ; \quad \Delta v_B^{(l)} = 0 \; ; \quad \Delta p_B^{(l)} = \Delta p_I^{(l)} \; .$$

The matrix $\mathcal{A}$ reads :

$$\mathcal{A} = \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 1 \end{bmatrix}$$

Incompressible outlet

We still consider the vector of the primitive variable :

$$\rho_B = \rho_I \; ; \quad u_B = u_I \; ; \quad v_B = v_I \; ; \quad p_B = p_\infty \; .$$

or

$$\Delta\rho_B^{(l)} = \Delta\rho_I^{(l)} \ ; \quad \Delta u_B^{(l)} = \Delta u_I^{(l)} \ ; \quad \Delta v_B^{(l)} = \Delta v_I^{(l)} \ ; \quad \Delta p_B^{(l)} = 0 \ .$$

The matrix $\mathcal{A}$ reads :

$$\mathcal{A} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 0 \end{bmatrix}$$

Low-Mach injection

In the case of a low-Mach injection we impose the inlet temperature, the inlet momentum, the tangential velocity and we extrapolate the pressure from the interior of the domain :

$$T_B = T_{\text{inj}} \ ; \quad \dot{m}_B = \dot{m}_{\text{inj}} \ ; \quad (V_t)_B = 0 \ ; \quad p_B = p_I \ .$$

If the vector q is such as $q = (T,\ \dot{m},\ V_t,\ p)^T$ the matrix reads :

$$\mathcal{A} = \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 1 \end{bmatrix}$$

Then, the primitive variables are computed as follows :

$$\begin{cases} \Delta p_B^{(l)} = \Delta p_I^{(l)} \ ; \\ \Delta\rho_B^{(l)} = \dfrac{1}{RT_{\text{inj}}} \Delta p_I^{(l)} \ ; \\ \Delta u_B^{(l)} = n_x \cdot R \cdot T_{\text{inj}} \cdot \dot{m}_{\text{inj}} \cdot \left(\dfrac{1}{p_I^{(l)}} - \dfrac{1}{p_I^n} \right) \ ; \\ \Delta v_B^{(l)} = n_y \cdot R \cdot T_{\text{inj}} \cdot \dot{m}_{\text{inj}} \cdot \left(\dfrac{1}{p_I^{(l)}} - \dfrac{1}{p_I^n} \right) \ . \end{cases}$$

Chapter 5

Stability Analysis of the Matrix-Free Method for the Euler equations

5.1 Introduction

The previous chapter has described a matrix-free implicit treatment that can be applied for all-speed flows with a particularly low cost per iteration. Let us emphasize again the different stages involved in the derivation of this implicit treatment :

- the primary form of the matrix-free implicit scheme is given by (4.2.7)-(4.2.8) and may be viewed as a first-order Rusanov-type implicit stage coupled with whatever high-order upwind explicit stage one may choose to use for accuracy purpose.

$$C_0 \Delta w_{i,j}^n \; + \; C_1^- \Delta w_{i-1,j}^n + \; C_1^+ \Delta w_{i+1,j}^n + \; C_2^- \Delta w_{i,j-1}^n + \; C_2^+ \Delta w_{i,j+1}^n$$

$$= \; -\Delta t \cdot \mathcal{R}_{i,j}^n \; - \; \sigma_1 \delta_1 \mu_1 \Delta (f^E)_{i,j}^n \; - \; \sigma_2 \delta_2 \mu_2 \Delta (g^E)_{i,j}^n$$

  with $\mathcal{R}$ the explicit residual and the scalar implicit coefficients given by :

$$C_1^{\pm} = -\tfrac{1}{2}\sigma_1 \left(\rho_1^E\right)_{i\pm\frac{1}{2},j}^n \quad , \quad C_2^{\pm} = -\tfrac{1}{2}\sigma_2 \left(\rho_2^E\right)_{i,j\pm\frac{1}{2}}^n \quad , \quad C_0 \; = 1 - C_1^- - C_1^+ - C_2^- - C_2^+ .$$

  As far as the amplification factor analysis of this scheme is concerned, the only difference with a block implicit treatment lies in the replacement of the matrix dissipation coefficients with the spectral radii ρ_1^E, ρ_2^E since in the linear (or linearized) case, where $f^E = A^E \cdot w$ with $A^E = constant$ (and similarly for g^E) one can write $\delta_1\mu_1\Delta(f^E) = A^E\delta_1\mu_1\Delta w^n$ (and similarly in the y direction). The use of $\delta_1\mu_1\Delta f^E$, $\delta_2\mu_2\Delta g^E$ for practical purpose allows to simplify and thus reduce the cost of the evaluation of these implicit terms, but bears no consequence on the scheme stability and efficiency with respect to the case of the block-implicit treatment relying on $A^E\delta_1\mu_1\Delta w^n$, $B^E\delta_2\mu_2\Delta w^n$.

- in a second step, the previous implicit stage must be approximately solved and this task is performed using Point-Jacobi relaxation method or the SGS relaxation method; both treatments allow to deal with the solution of the above implicit stage including variable and flux vector increments (Δw^n, $\Delta(f^E)^n$, $\Delta(g^E)^n$) : they are summarized by formulae (4.2.12) and (4.2.13) given in the previous chapter.

- in a third and last step, the PJ-MF and SGS-MF treatments are expressed in the form (4.3.15) so as to minimize their unit computational cost when low-Mach preconditioning of the governing equations is applied.

In this chapter, we want first to assess specifically the impact on intrinsic efficiency of the choice of a first-order Rusanov-type implicit stage coupled with a third-order Roe explicit stage, instead of a consistent first-order Roe (block-)implicit stage as done in chapter 1 to 3 of this thesis. In other words, taking into account the above remarks, we simply want to assess the loss of intrinsic efficiency introduced by the spectral radius simplification on the numerical dissipation in the implicit stage. We focus on the steady Euler equations; Navier-Stokes and unsteady analyses will be successively presented in chapters 6 and 7. Next, we also analyze the implicit Point Jacobi and Symmetric Gauss-Seidel treatments applied to the matrix-free scheme.
The Von Neumann analysis for the linearized Euler equations is performed in the manner of the third chapter. More precisely, it was shown that inviscid Jacobians and dissipation matrices could be expressed using a limited number of parameters such as the flow angle, the Mach number, the CFL number and the aspect ratio $\delta y/\delta x$ (*cf.* section 3.5).

5.2 Impact of the Matrix-Free simplification

5.2.1 Global analysis

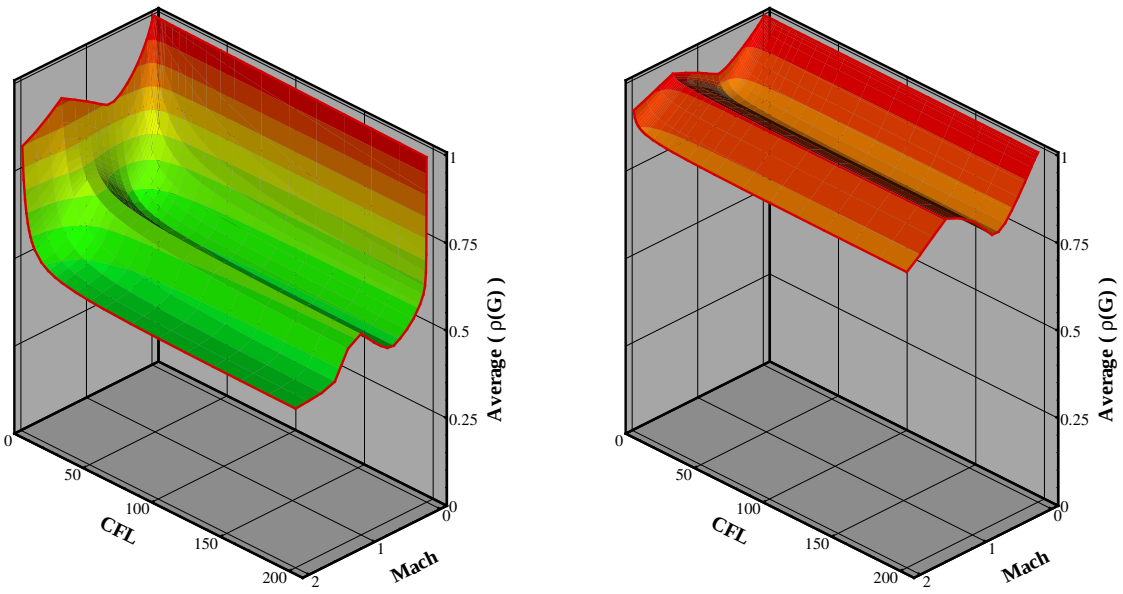


Figure 5.2.1: Average amplification factor of the Direct scheme. Left : Block scheme. Right : Matrix-Free scheme.

In the manner of the third chapter, we carry out a global analysis with respect to the Mach number and the CFL number. The results for the non-preconditioned scheme are presented in figure 5.2.1. We note immediately that the matrix-free simplification reduces dramatically the efficiency of the implicit scheme, especially for the Mach number close to unity and for the very

low Mach numbers. We also remark that, unlike block scheme, high CFL values do not enable to improve the average damping. The results for the preconditioned scheme are presented in figure 5.2.2. As expected, the low-Mach preconditioner enhances the intrinsic efficiency over the low Mach numbers domain, and it is also remarkable that the loss of efficiency due to the simplification is less important than for higher Mach numbers. Such results reveal that the method requires a large amount of iteration to reach the steady state and, consequently, the cost per iteration of the Matrix-Free method has to be very cheap in order to recover a competitive global cost. The study of the cost per iteration is the subject of the third part of this thesis. In the following paragraph, we will try to explain why the matrix-free simplification is less penalizing over the low-Mach domain and we will detail more precisely the impact of the preconditioning on the matrix-free scheme.

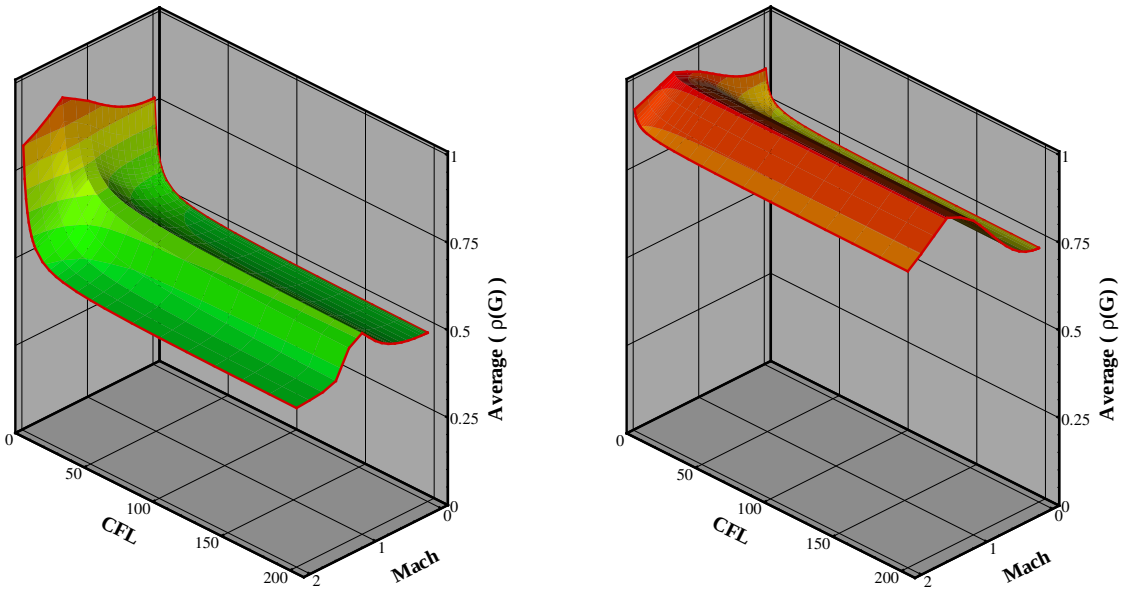


Figure 5.2.2: Average amplification factor of the preconditioned Direct scheme. Left : Block scheme. Right : Matrix-Free scheme.

5.2.2 Local analysis

In order to carry out a precise analysis of the properties of the Matrix-Free method, we use a local analysis. We showed in chapter 3 that using high CFL numbers enable to obtain a good damping over the low-frequency domain. We therefore choose such values to perform our analyses.

We first set the Mach number at $M = 0.5$ and the CFL number at $CFL = 10^6$, then we draw the amplification factor map for the Matrix-Free scheme with and without low-Mach preconditioner. The results are exhibited in figure 5.2.3. There are slightly differences between both methods but the preconditioned scheme seems to provide a slight better damping over the high frequency region. Nonetheless, the global efficiencies are very similar.

Second we set the Mach number at $M = 0.1$, the results are presented in figure 5.2.4.

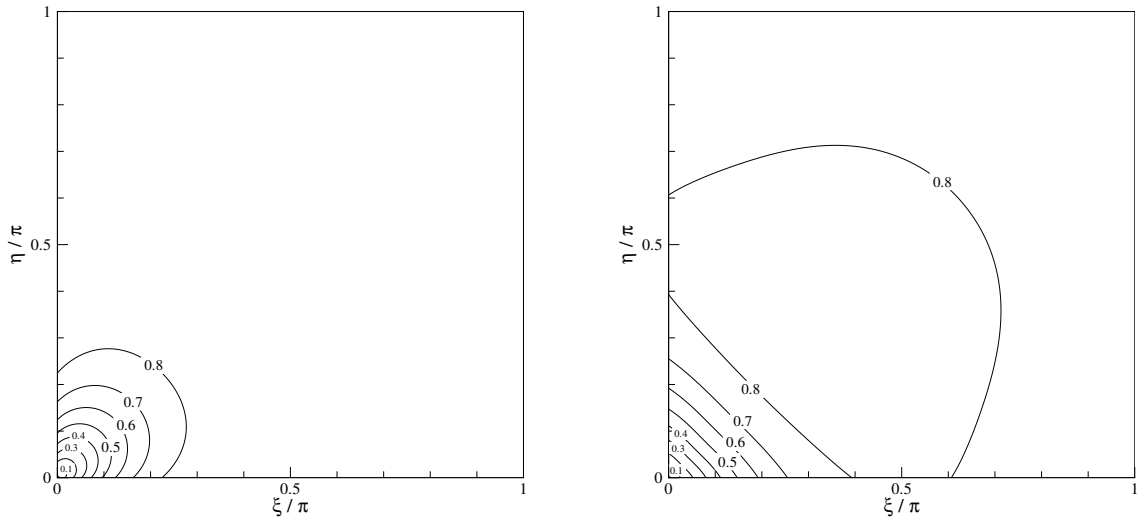


Figure 5.2.3: Amplification factor map of the Matrix-Free Direct scheme at Mach $M = 0.5$ and $CFL = 10^6$. Left : No preconditioning. Right : Turkel preconditioning.

The non-preconditioned scheme is highly inefficient over a large part of the frequency domain showing that it is not worthwhile to use such a method to compute low-Mach number flows. On the other hand, the preconditioned scheme provides a very good damping and, as forecasted by the global analysis, its efficiency is better than for higher Mach numbers. In order to emphasize this property, we plot the amplification factor of both methods along the line $\xi = \eta$ at the same Mach numbers and we compare them with respect to the block scheme's damping. The results are exhibited in figure 5.2.5. We observe that the gap between the Matrix-Free method and the Block scheme is reduced for the small Mach number. A potential explanation of such results can be found in the effects of the preconditioner. Indeed, the goal of preconditioning techniques is to make the condition number of the system close to unity. It means that the gap between all the eigenvalues is reduced dramatically for the low Mach limit. Moreover, the efficiency of the Turkel preconditioner, for instance, is better when Mach number is smaller than 0.1. As a consequence, the impact of maximizing the numerical dissipation, as it is the case for the Matrix-Free scheme, is less important for low-Mach numbers for which the eigenvalues are very close than for higher Mach values.

To achieve the analysis of the impact of the spectral radius simplification, we draw the amplification map of Block and Matrix-Free schemes at Mach $M = 10^{-3}$. We notice that the shapes of the amplification factor maps are similar for both methods. Matrix-Free scheme provides a good damping over the very low-frequency domain, but its efficiency is very low for the rest of the frequency map. Nevertheless, the gap between both methods is not so important, and, although it requires more iteration to damp the error modes, Matrix-Free method can be competitive if its cost per iteration is very cheap.

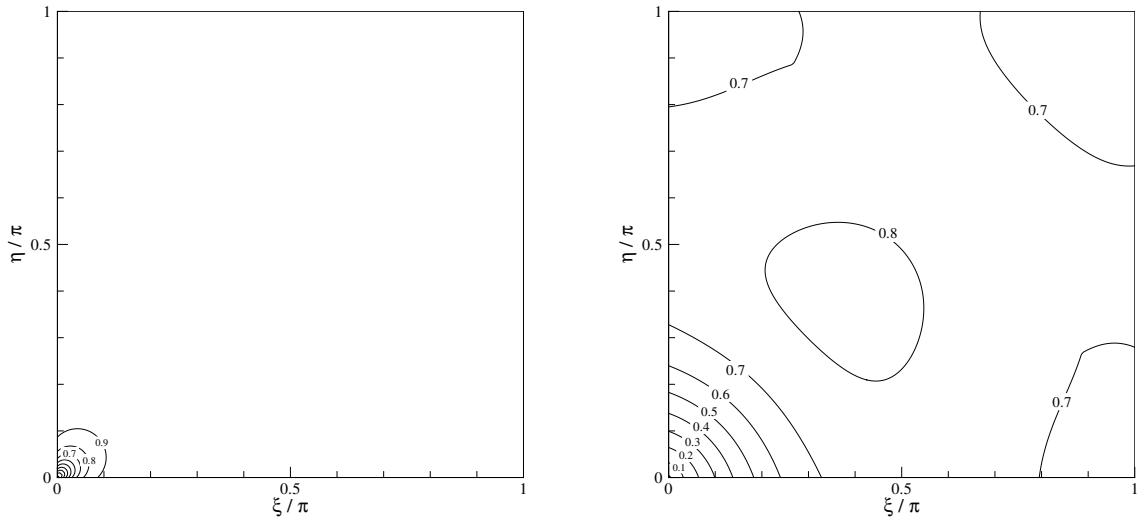


Figure 5.2.4: Amplification factor map of the Matrix-Free Direct scheme at Mach $M = 0.1$ and $CFL = 10^6$. Left : No preconditioning. Right : Turkel preconditioning.

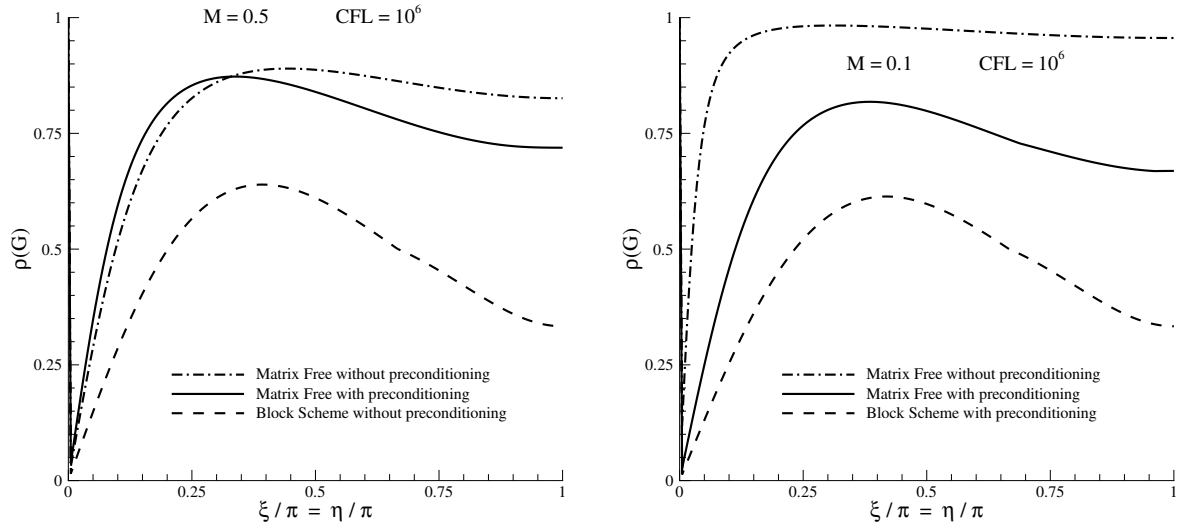


Figure 5.2.5: Amplification factor along the axis $\xi = \eta$ at $CFL = 10^6$. Left : Mach $M = 0.5$. Right : Mach $M = 0.1$.

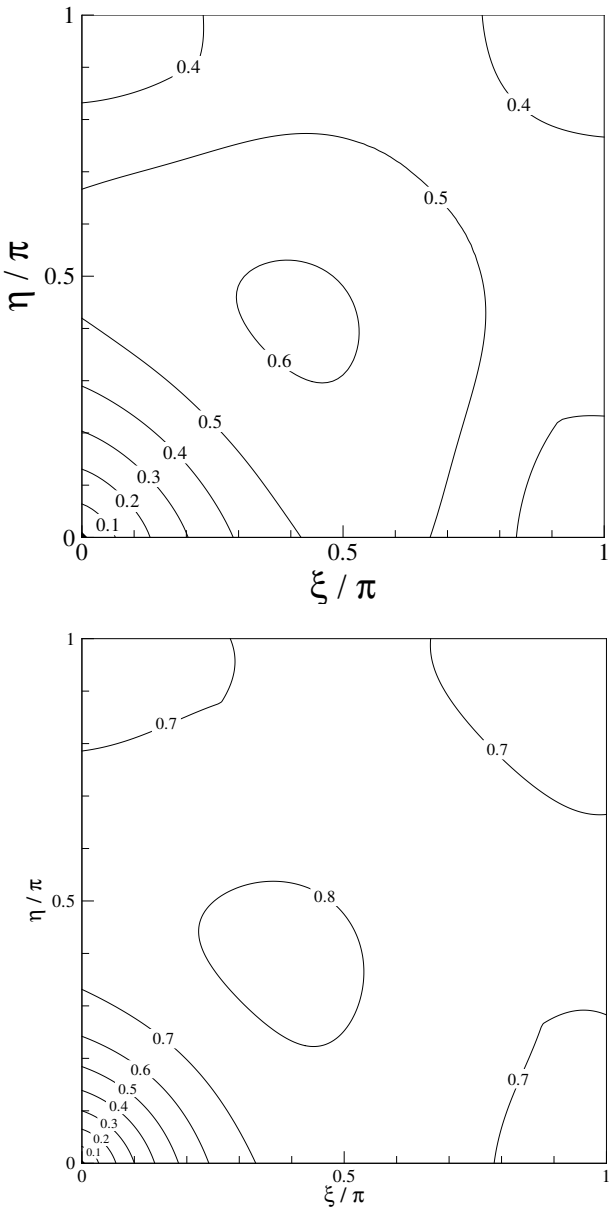


Figure 5.2.6: Amplification factor map for the Direct scheme at Mach $M = 10^{-3}$ and $CFL = 10^6$. Top : Block scheme. Bottom : Matrix-Free scheme.

5.3 Matrix-Free Symmetric Gauss-Seidel scheme

In chapter 3, we studied the SGS scheme applied to an implicit Block stage and we outlined that, unless performing several inner iterations, the method was unstable for high CFL numbers. However, in the purpose of computing flows on unstructured meshes, SGS procedure is very easy to implement. In this section, we want to know if such a procedure is still interesting coupling with a matrix-free implicit stage. We lead global and local analyses in order to come into conclusion.

5.3.1 Global analysis

The maximum and the average values of the amplification factor versus CFL and Mach numbers are presented in figure 5.3.1 for one inner iteration and in figure 5.3.2 for 20 inner iterations. Only one inner iteration is not sufficient to provide stability for all the Mach numbers. However, using 20 inner iterations does not really increase the cost per iteration of the MF-SGS method but it enables to recover the performances of the Direct scheme. Such a method is therefore promising in terms of cost and efficiency.

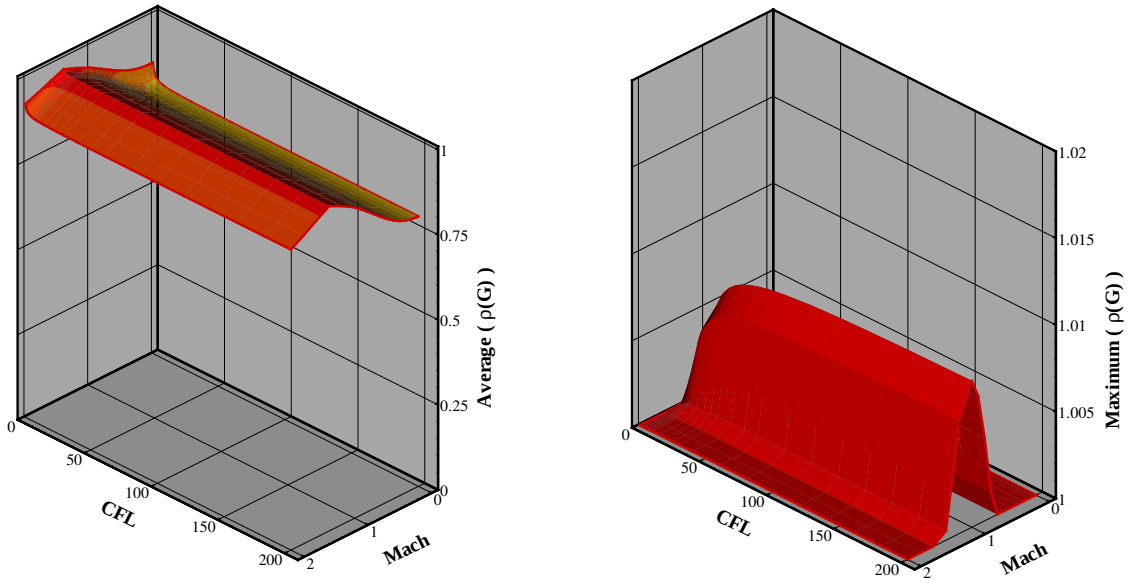


Figure 5.3.1: Amplification factor of the MF-SGS(1) scheme. Left : Average. Right : Maximum.

5.3.2 Local analysis

The local analysis is made for a low-Mach number case ($M = 10^{-3}$) and at $CFL = 10^6$. In such conditions, the MF-SGS scheme is stable but its efficiency is poor, specially in the low wavenumber regions (*cf.* figure 5.3.3). Using 30 inner-iterations enables to recover the efficiency is very close to the Direct Scheme's one. Thanks to its simplicity and its good convergence properties, SGS relaxation procedure is well-suited to be coupled with the Matrix-Free implicit stage.

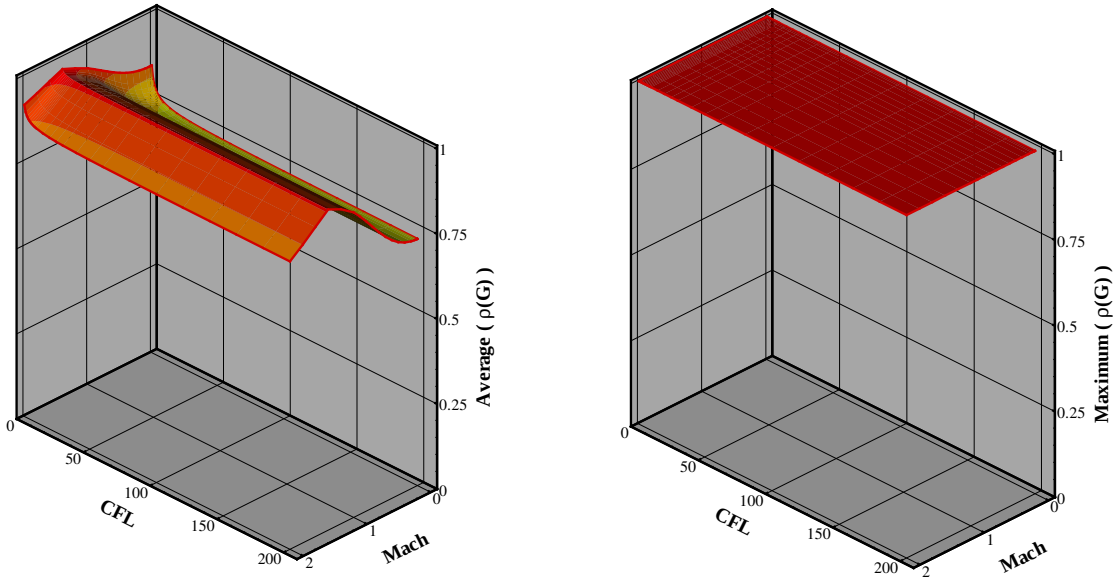


Figure 5.3.2: Amplification factor of the MF-SGS(20) scheme. Left : Average. Right : Maximum.

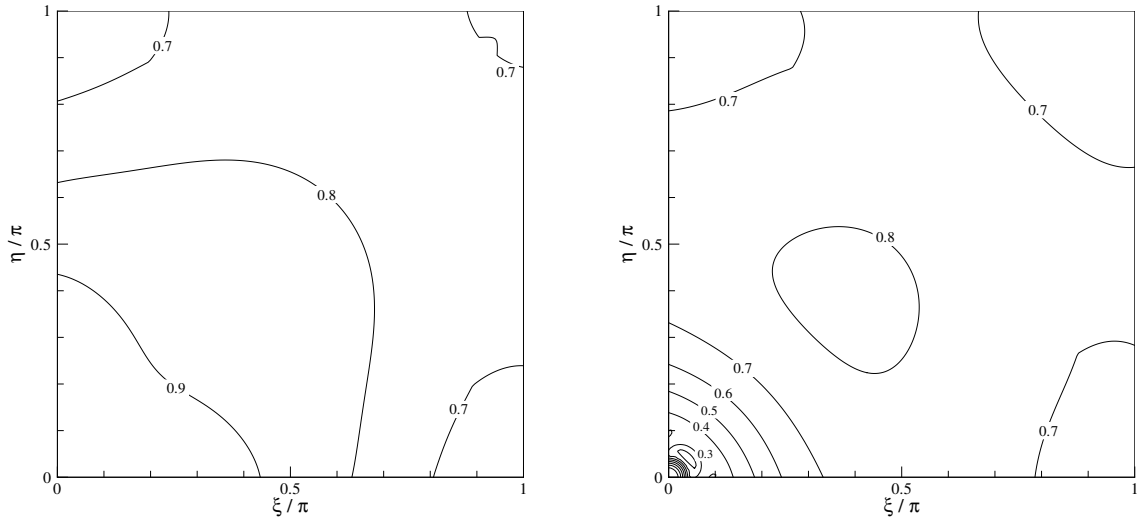


Figure 5.3.3: Amplification factor at $M = 10^{-3}$ and $CFL = 10^{6}$. Left : MF-SGS(1) scheme. Right : MF-SGS(30).

5.4 Matrix-Free Point Jacobi scheme

In chapter 3, Point Jacobi Block scheme was shown to be very unstable and ineffective. However, practical computations using MF-PJ method has been carried out for several years in SINUMEF laboratory. In our first developments (*cf.* [26]), we used with success the same technique to compute low-Mach number flows. The properties of this method were obviously not well-known. Thus, it appeared necessary to investigate further the performances of such a method.

5.4.1 Global analysis

MF-PJ scheme with only one sub-iteration is strongly unstable for all CFL numbers and for all Mach numbers (*cf.* figure 5.4.1). However, unlike PJ Block scheme, 50 inner iterations enable to provide stability even for large CFL numbers (*cf.* figure 5.4.2). Besides, the plot of the average amplification factor shows that the efficiency is very similar to the MF-SGS method's one. In the next paragraph, we will study more precisely the efficiency of the MF-PJ method at very great CFL number.

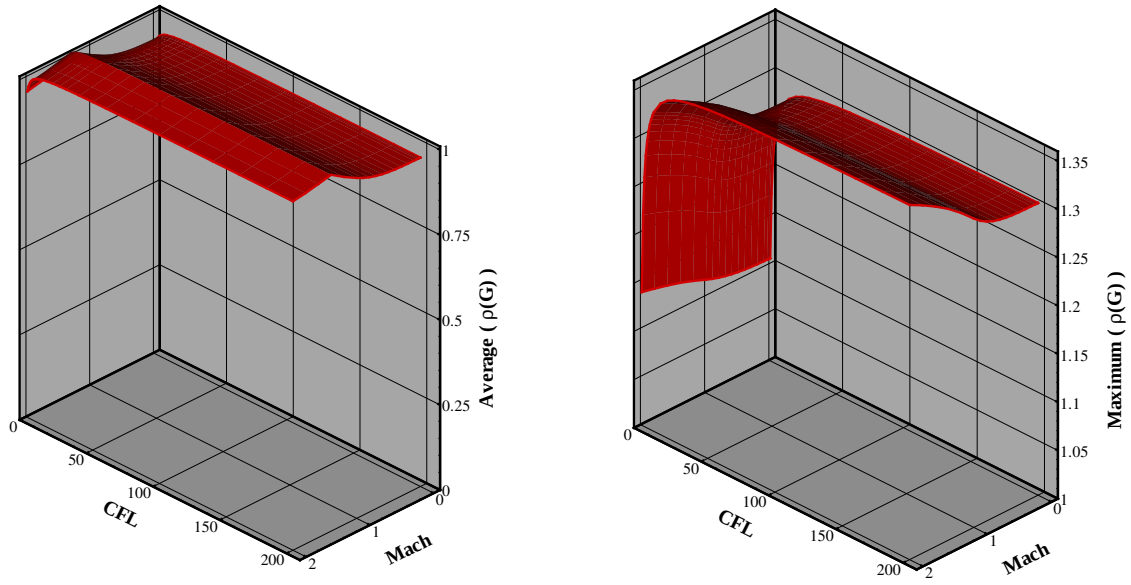


Figure 5.4.1: Amplification factor of the MF-PJ(1) scheme. Left : Average. Right : Maximum.

5.4.2 Local analysis

The first local analysis is made at Mach $M = 10^{-3}$ and at CFL 10^6. The results in figure 5.4.3 emphasize the amplification factor for one and 50 inner iterations. MF-PJ(1) is strongly unstable but this issue can be partially circumvented using MF-PJ(50). Indeed, instabilities are pinpointed in the very low-frequency region, in other words, the method would be unstable for fine meshes only. Two strategies are then imaginable, namely reducing the CFL number at the expense of a great loss of efficiency for the low-frequency modes, or increasing the number of sub-iterations to preserve the intrinsic efficiency at the expense of the global cost. Figure 5.4.4 presents the amplification factor maps of MF-PJ(50) at $CFL = 10^2$ and of MF-PJ(500)

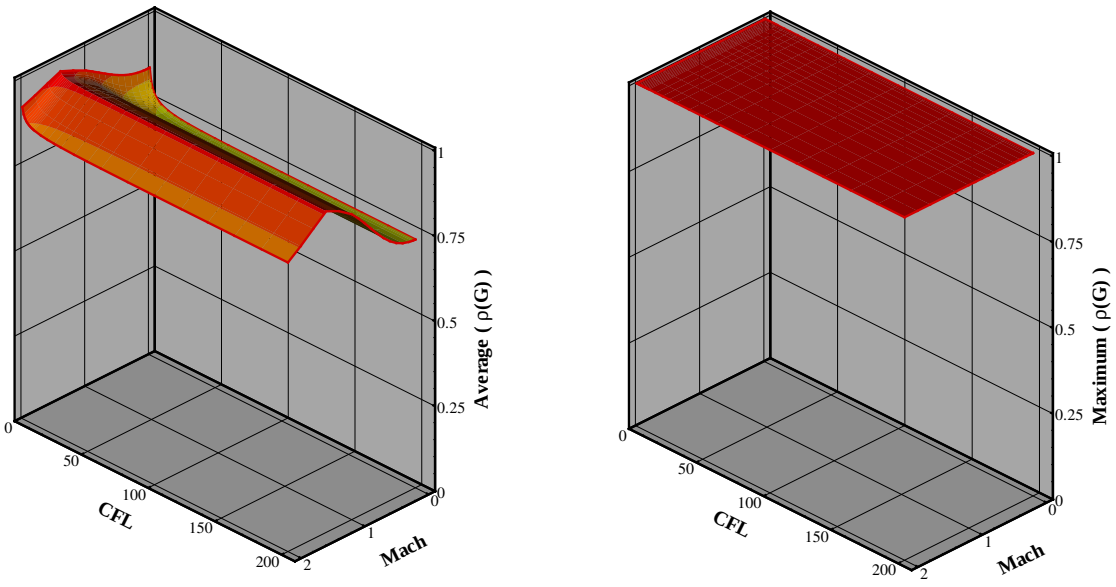


Figure 5.4.2: Amplification factor of the MF-PJ(50) scheme. Left : Average. Right : Maximum.

at $CFL = 10^6$. Both methods are stable and their damping properties are very similar except for the very low-frequency area. The final choice relies on subjective considerations, in our case, we favor high CFL numbers and, consequently, a more important amount of inner iterations.

SGS and PJ relaxation procedures seem to be valuable choices to solve an implicit Matrix-Free stage. They both provide simplicity and efficiency compare to the Matrix-Free Direct scheme. Nonetheless, we need further analyses to know if such methods are competitive in all cases.

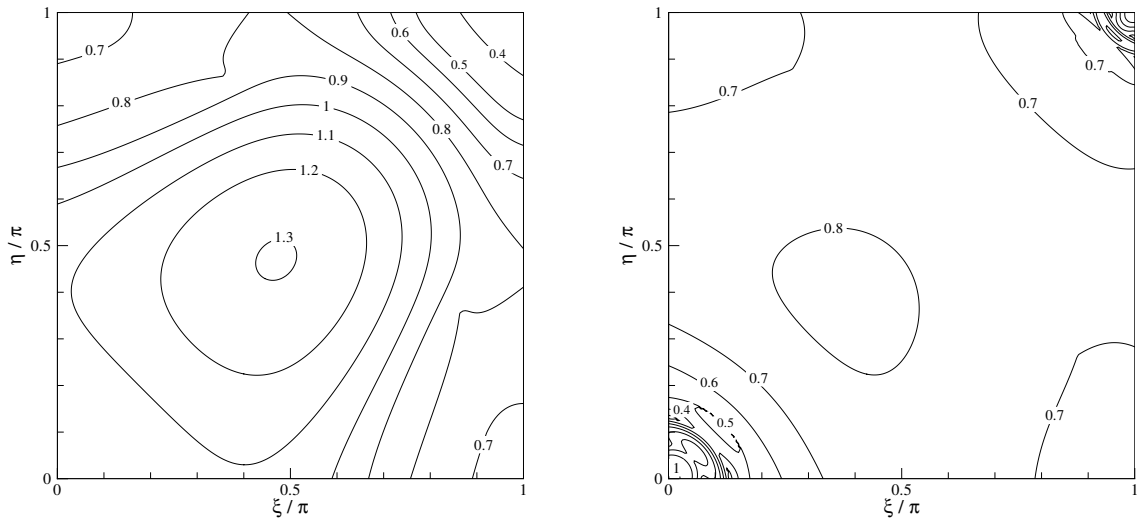


Figure 5.4.3: Amplification factor at $M = 10^{-3}$ and $CFL = 10^6$. Left : MF-PJ(1) scheme. Right : MF-PJ(50).

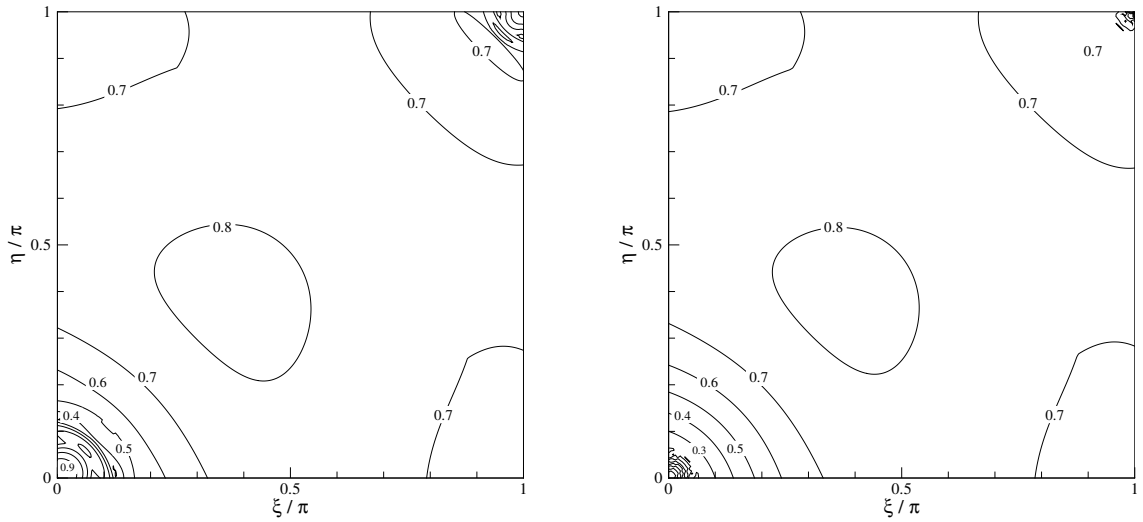


Figure 5.4.4: Amplification factor at $M = 10^{-3}$. Left : MF-PJ(50) scheme at $CFL = 10^2$. Right : MF-PJ(500) scheme at $CFL = 10^6$.

5.5 Effect of high aspect ratio

We now turn our attention to the relative performance of the Matrix-Free implicit method in the presence of high aspect ratio grids. Several authors have studied the effects of high aspect ratio on Block scheme solved by ADI method [11] or by relaxation procedures [12]. An interesting definition of the time step is given by Buelow *et al.*, namely the "min-CFL" definition for which we have :

$$\Delta t = \max\left[\frac{CFL \cdot \delta x}{\lambda_1^+}, \frac{CFL \cdot \delta y}{\lambda_2^+}\right] \tag{5.5.1}$$

This definition enables to maintain an optimum CFL value in the axial direction in the case of high aspect ratio. However, the authors show also that the SGS Block scheme looses a lot of efficiency at high aspect ratio and this definition does not improve the scheme. In our case, we want to know whether the same behaviour is observed with the MF-SGS scheme and the MF-PJ scheme.

5.5.1 MF-SGS scheme

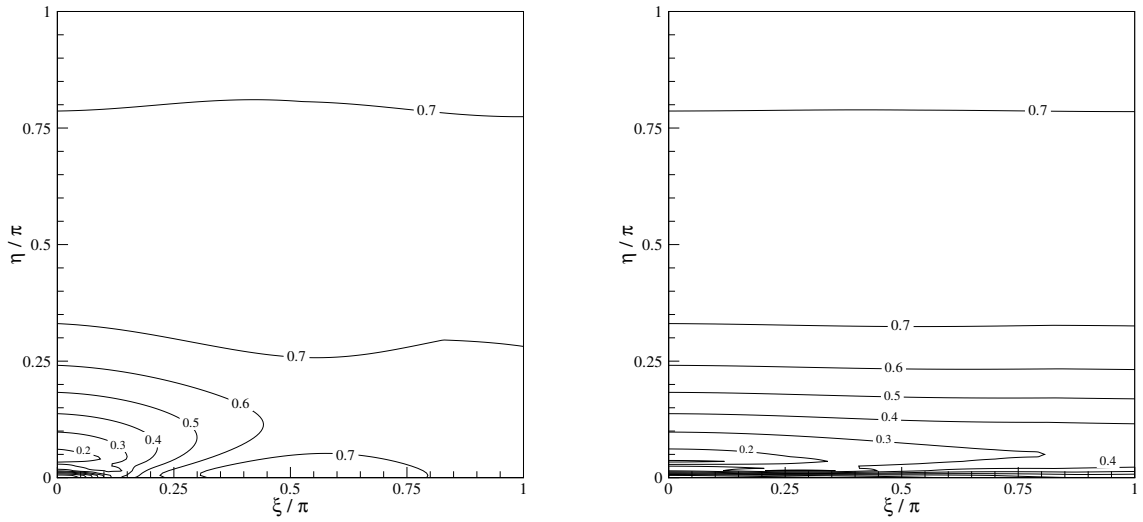


Figure 5.5.1: Amplification factor map of the MF-SGS(40) scheme at $M = 10^{-3}$ and $CFL = 10^3$. Left : Aspect ratio of 10^{-1}. Right : Aspect ratio of 10^{-2}.

In figure 5.5.1, we plotted the amplification factor map of the MF-SGS(40) scheme at $CFL = 10^3$ and $M = 10^{-3}$ and for two different values of the aspect ratio, namely 10^{-1} and 10^{-2}. We use the original CFL definition for the moment. As long as the aspect ratio decreases, the amplification factor of the MF-SGS method deteriorates for the pure axial mode, that is for $\eta = 0$. We can visualize more clearly such a behaviour in the figure 5.5.2 which represents the profile of the amplification factor along the x-axis. We compare the MF-SGS scheme to the MF Direct Scheme, and we notice that the damping of MF-SGS method is reduced dramatically as soon as the aspect ratio is lower than 10^{-1}. MF-SGS method is well conditioned for the mid-wavenumbers but it presents a large lack of efficiency for the purely longitudinal modes, indicating poor convergence

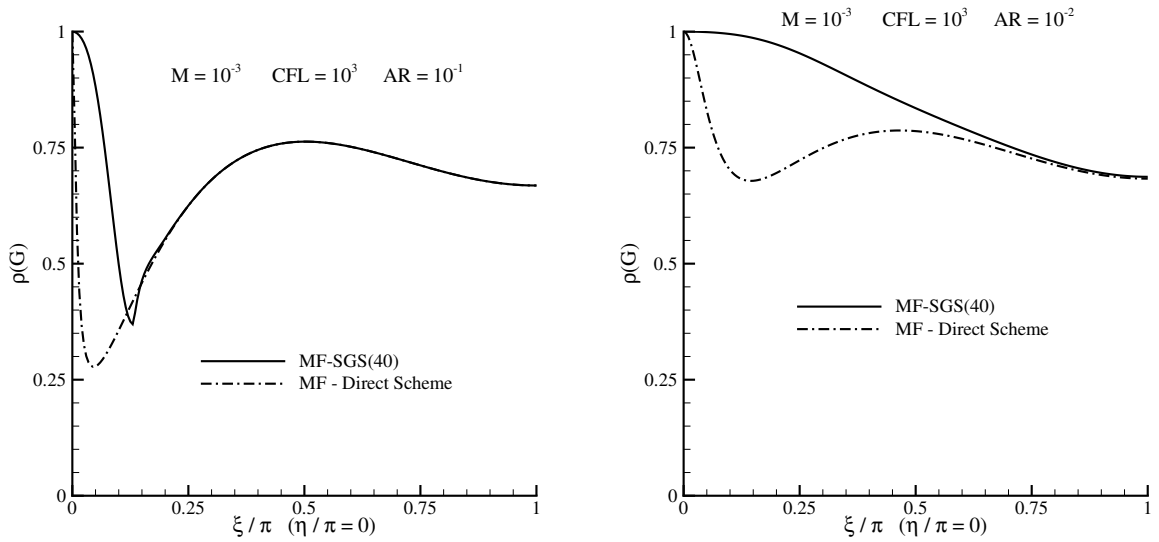


Figure 5.5.2: Amplification factor profile along x-axis at $M = 10^{-3}$ and $CFL = 10^3$. Left : Aspect ratio of 10^{-1}. Right : Aspect ratio of 10^{-2}.

rate.
In order to circumvent this issue, we employ the "min-CFL" definition and we draw the x-profile of the amplification factor at $AR = 10^{-4}$. Results are presented in figure 5.5.3. The "min-CFL" definition enables the Direct Scheme to recover the efficiency of $AR = 1$, unfortunately, it does not improve the damping of the MF-SGS method which remains stiff for axial modes. According to this study, we can conclude that the matrix-free simplification is not the cause of the loss of efficiency at high aspect ratio. The stiffness of the method for purely axial modes relies on the SGS algorithm which is unsuited to high aspect ratio grids. We will study in the next chapter whether the same tendency is still valid for Navier-Stokes equations. However, we can already mention that this is a weakness of the method.

5.5.2 MF-PJ scheme

To our knowledge, effects of high aspect ratio on PJ procedure have not been studied. Hence, it is interesting to study the behaviour of this method for such conditions. High aspect ratio have surprising effects on the MF-PJ method. We recall that PJ relaxation procedure could lead to oscillation in the damping, and sizes of such oscillations could cause instabilities. However, we showed that, coupled with an implicit matrix-free stage, the algorithm remained stable. Setting the aspect ratio at 10^{-1} creates oscillations in the low-frequency region and the high-frequency area (*cf.* figure 5.5.4 left part). The oscillating phenomena spreads to the whole wavenumber domain for higher aspect ratio (*cf.* figure 5.5.4 right part), leading to strong instability.
For aspect ratio around 10^{-4}, it is necessary to reduce the CFL number in order to preserve stability. For such aspect ratio, the damping is almost unidirectional, thus we can easily visualize the efficiency of the method by plotting the amplification factor along the y-direction. In figure 5.5.5, we observe clearly that the oscillating phenomena cause instability for great CFL numbers, and we notice that using a great amount of inner iteration does not circumvent the problem (right part of the figure). Nonetheless, this phenomena can be mitigated by setting a more reasonable CFL number, MF-PJ method is then stable and quite efficient (left part of the figure) for purely

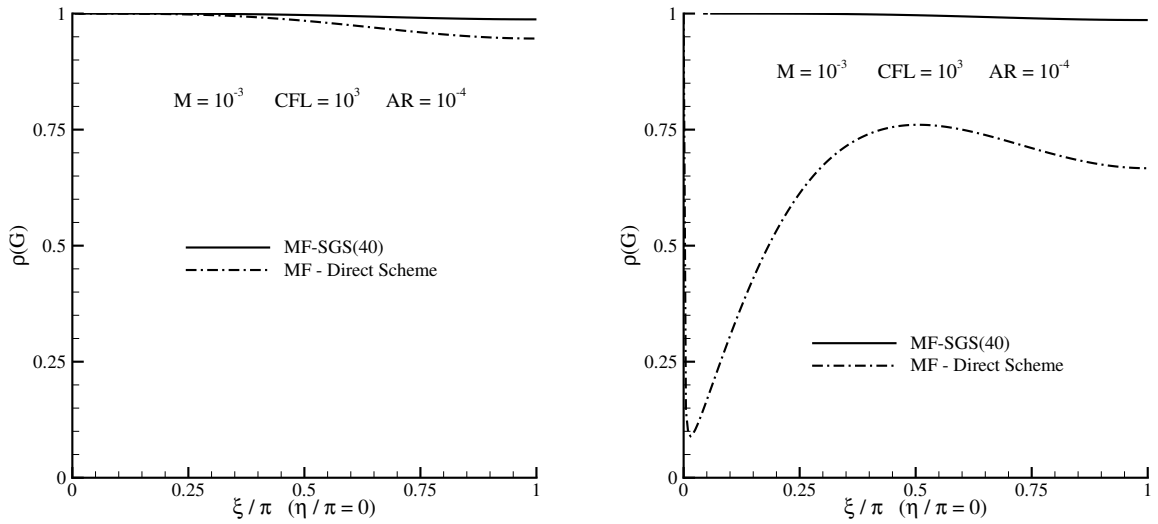


Figure 5.5.3: Amplification factor at $M = 10^{-3}$, $CFL = 10^6$ and $AR = 10^{-4}$. Left : Whole wavenumber domain. Right : Close-up of region near x-axis.

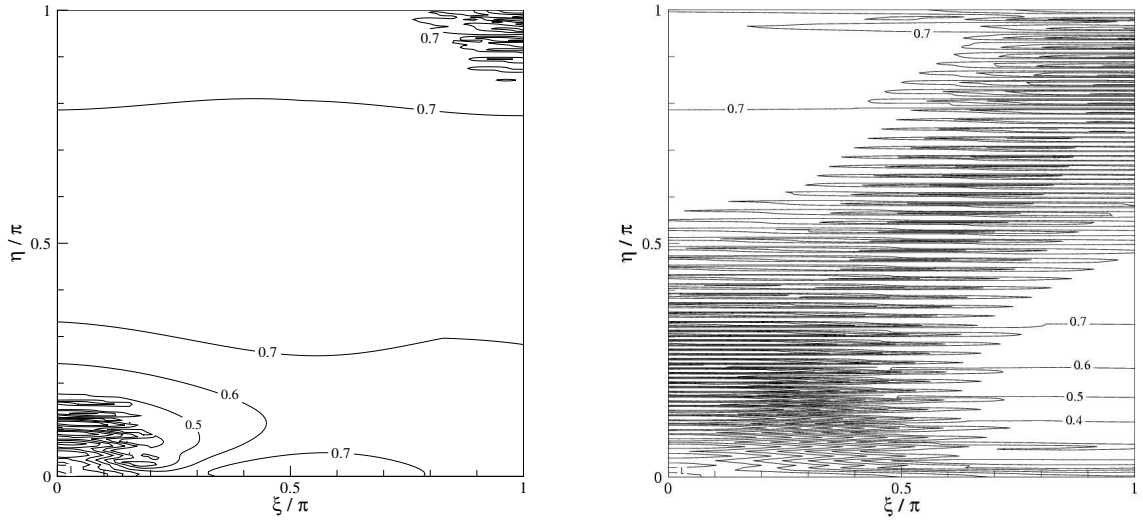


Figure 5.5.4: Amplification factor map of the MF-PJ(50) scheme at $M = 10^{-3}$ and $CFL = 10^6$. Left : $AR = \delta y/\delta x = 10^{-1}$. Right : $AR = 10^{-2}$.

cross-stream modes. Unfortunately, MF-PJ method presents the same drawbacks than the MF-SGS technique, that is a very poor damping of longitudinal disturbances (not shown here since the results are identical to the ones obtained with MF-SGS method). PJ algorithm, in the manner of the SGS algorithm, appears to be unsuited to high aspect ratio grids. Next chapter will deal with the case of the Navier-Stokes equations so as to confirm such a behaviour.

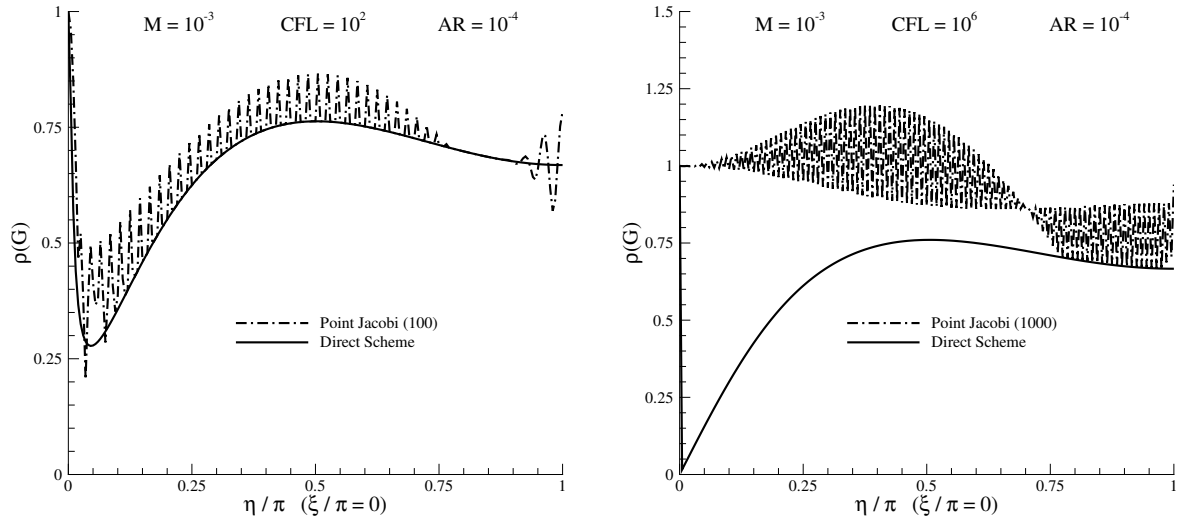


Figure 5.5.5: Profile of the amplification factor along the y-axis at $M = 10^{-3}$ and $AR = 10^{-4}$. Left : PJ(100) scheme at $CFL = 100$. Right : PJ(1000) scheme at $CFL = 10^6$.

5.6 Conclusions

In this chapter, we saw that the spectral radius simplification reduces the damping properties of the implicit Matrix-Free scheme. However, such a simplification is less penalizing at low-Mach number. Thus this loss of intrinsic efficiency does not affect the global cost of the method providing that its cost per iteration is very low. In chapter 4, we saw that SGS and PJ algorithm ensure the simplicity and the low cost of the Matrix-Free method. Besides, the studies that we lead in this chapter show that such algorithms are well-suited to solve Euler equations since they provide a good damping for a reasonable amount of inner-iterations. However, these algorithms make the method stiff for purely axial modes when high aspect ratio grids are used. These schemes are therefore less attractive as soon as we have to deal with very stretched grids. In the next chapter, we will study whether it is still the case for the Navier-Stokes equations.

Chapter 6

Stability Analysis of the Matrix-Free Method for the Navier-Stokes equations

6.1 Introduction

In the previous chapter, the impact of the matrix-free simplification on the intrinsic efficiency has been studied in the case of the Euler equations. We now want to evaluate the loss of intrinsic efficiency due to the viscous spectral radius simplification in the implicit stage. We recall that for the Navier-Stokes equations, the form of the Matrix-Free scheme is formally unchanged with respect to the inviscid case :

$$C_0\Delta w^n_{i,j} \; + \; C_1^-\Delta w^n_{i-1,j} + \; C_1^+\Delta w^n_{i+1,j} + \; C_2^-\Delta w^n_{i,j-1} + \; C_2^+\Delta w^n_{i,j+1}$$
$$= \; -\Delta t\cdot\mathcal{R}^n_{i,j} \; - \; \sigma_1\delta_1\mu_1\Delta(f^E)^n_{i,j} \; - \; \sigma_2\delta_2\mu_2\Delta(g^E)^n_{i,j}$$

with $\mathcal{R}$ the explicit residual involving the viscous contribution and the scalar implicit coefficients now given by :

$$C_1^{\pm} = -\frac{1}{2}\sigma_1\left(\rho_1^E+\rho_1^V\right)^n_{i\pm\frac{1}{2},j} \; , \; C_2^{\pm} = -\frac{1}{2}\sigma_2\left(\rho_2^E+\rho_2^V\right)^n_{i,j\pm\frac{1}{2}} \; , \; C_0 \; = 1-C_1^- - C_1^+ - C_2^- - C_2^+.$$

When the Matrix-Free scheme is applied to solve viscous low-Mach number flows, the preconditioning matrix P appears in the previous coefficients (*cf.* expression (4.2.23)). In this chapter, the impact of the viscous spectral radius simplification will be first studied, then we will also analyse the efficiency of the viscous preconditioning for the Matrix-Free approach. Eventually, the performances of the MF-SGS and MF-PF schemes will be studied, especially in the case of high grid aspect ratio.

6.2 Parameters of the study

In the manner of the previous chapter, we need to define carefully the parameters of the study. After non-dimensionalization, inviscid Jacobians can be expressed thanks to the Mach number M, the flow angle v/u and the ratio of specific heats γ, while the viscous Jacobians require the

Reynolds and the Prandtl numbers in addition. Convective and diffusive phenomena possess their respective characteristic time-step which can be defined as follows :

$$\begin{cases} \Delta t^E = \min(\frac{\delta x}{\lambda_1^+}, \frac{\delta y}{\lambda_2^+}) \\ \Delta t^V = \min(\frac{\delta x^2}{\rho^V}, \frac{\delta y^2}{\rho^V}) \end{cases}$$

where λ_1^+ and λ_2^+ are the spectral radius of the eigensystem in each direction while ρ^V is the viscous spectral radius which, for the air, reads :

$$\rho^V = \frac{\gamma\mu}{PrRe\rho}$$

We can choose the time-step as the minimum between the convective and the diffusive characteristic time multiplied by a safety factor χ :

$$\Delta t = \chi \cdot \min(\Delta t^E, \Delta t^V)$$

When convective effects are dominant, typically far from the walls, we have $\Delta t^E << \Delta t^V$, the time-step is therefore given by :

$$\Delta t = \chi \cdot \Delta t^E$$

and the coefficient χ would correspond to the CFL number. On the other hand, we get $\Delta t^V << \Delta t^E$ when viscous effects are preponderant, that is in the shear layer, the time-step is then defined by :

$$\Delta t = \chi \cdot \Delta t^V$$

and the coefficient χ would correspond to the VNN number. In order to lead stability analyses for the Navier-Stokes equations, we need a significant parameter in order to determine whether the flow is dominated by the convective effects or whether it is dominated by the viscous effects. In each direction, one can define a cell Reynolds number as follows :

$$Re_x = \frac{u \cdot \delta x}{\rho^V} \quad , \qquad Re_y = \frac{v \cdot \delta y}{\rho^V}$$

The cell Reynolds number is then defined as the minimum between these two values :

$$Re_m = \min(Re_x, Re_y) = \frac{u \cdot \delta x}{\rho^V} \cdot \min(1, \frac{AR}{v/u})$$

If we suppose that the flow angle is equal to 45 degrees, then the ratio v/u is equal to unity and the cell Reynolds number reads :

$$Re_m = \frac{u \cdot \delta x}{\rho^V} \cdot \min(1, AR) \tag{6.2.1}$$

This cell Reynolds number enables us to evaluate the ratio between the viscous time-step and the convective time-step. Indeed, we have :

$$\frac{\Delta t^V}{\Delta t^E} = \frac{\lambda_1^+ \cdot \delta x}{\rho^V} \cdot \min(1, AR^2) \cdot \left[\min(1, \frac{AR}{\lambda_2^+/\lambda_1^+})\right]^{-1}$$

which, in the case of a 45 degree flow angle, reduces to :

$$\frac{\Delta t^V}{\Delta t^E} = \frac{\lambda_1^+ \cdot \delta x}{\rho^V} \cdot \min(1, AR) = \frac{\lambda_1^+}{u} \cdot Re_m = \Theta \tag{6.2.2}$$

The ratio between the two time-steps relies on the cell Reynolds number and the ratio between the biggest eigenvalue and the particle velocity. In the case of the inviscid preconditioning, the definition of the preconditioned eigenvalue yields :

$$\lambda_1^+ \approx u$$

which leads to :

$$\frac{\Delta t^V}{\Delta t^E} = Re_m$$

However, we will see that the definition of the preconditioning should be modify in order to remain effective for the viscous dominated flows. Such a modification makes the eigenvalue λ_1^+ different from the particle velocity. For this reason we will denote the ratio of the viscous time-step with respect to the convective time-step by Θ :

$$\Delta t = \chi \cdot \Delta t^E \cdot \min(1, \Theta) = \chi \cdot \Delta t^V \cdot \min(1, 1/\Theta)$$

Now we can easily determine all the Jacobians of the problem as follows :

$$\begin{cases} \dot{A}^E = \frac{\Delta t}{\delta x} A^E = \chi \cdot \min(\frac{1}{\rho(A^E)}, \frac{AR}{\rho(B^E)}) \cdot \min(1, \Theta) \cdot A^E(v/u, M, \gamma) \\ \dot{B}^E = \frac{\Delta t}{\delta y} B^E = \chi \cdot \min(\frac{1/AR}{\rho(A^E)}, \frac{1}{\rho(B^E)}) \cdot \min(1, \Theta) \cdot B^E(v/u, M, \gamma) \\ \dot{A}^V = \frac{\Delta t}{\delta x^2} A^V = \chi \cdot \frac{Pr}{\gamma} \cdot \min(1, AR^2) \cdot \min(1, 1/\Theta) \cdot \bar{A}^V(v/u, M, \gamma, Pr) \\ \dot{B}^V = \frac{\Delta t}{\delta y^2} B^V = \chi \cdot \frac{Pr}{\gamma} \cdot \min(1, 1/AR^2) \cdot \min(1, 1/\Theta) \cdot \bar{B}^V(v/u, M, \gamma, Pr) \\ \dot{S}^V = \frac{\Delta t}{\delta x \delta y} S^V = \frac{1}{3} \cdot \chi \cdot \frac{Pr}{\gamma} \cdot \min(AR, 1/AR) \cdot \min(1, 1/\Theta) \cdot \bar{S}^V(v/u, M) \end{cases} \tag{6.2.3}$$

where $\bar{A}^V$, $\bar{B}^V$ and $\bar{S}^V$ are given in appendix.

Finally, the required parameters which define the whole viscous problem are the following : v/u, M, γ, χ, AR, Pr and Re_m.

6.2.1 Low-Mach parameter

An additional parameter is necessary to carry out studies at small Mach number, namely the low-Mach parameter β that we introduced in chapter 1. For inviscid flows, it is customary to take it equal to the Mach number :

$$\beta^2 = \min[M^2, 1]$$

However, such a definition was shown to be unadapted to viscous flows [14], specially when the Reynolds number is very small. In order to emphasize the issue due to the use of the inviscid

definition, let us consider a low-Mach viscous dominated flow at $M << 1$ and $Re_m << 1$ and let us study the CFL and the VNN numbers :

$$CFL_x = \frac{\lambda_1^+ \cdot \Delta t}{\delta x}$$

$$VNN_x = \frac{\rho^V \cdot \Delta t}{\delta x^2}$$

Using the inviscid preconditioning yields $\lambda_1^+ \approx u$ and as $Re_m << 1$ it follows that :

$$CFL_x = \frac{\lambda_1^+ \cdot \Delta t}{\delta x} \approx \frac{u \cdot \Delta t}{\delta x} << \frac{\rho^V \cdot \Delta t}{\delta x^2} = VNN_x$$

For viscous dominated flows, the parameter χ enables to optimize the VNN number only. The inviscid preconditioning does not ensure that the CFL number would be set at its optimum value. In order to circumvent this issue, Venkateswaran *et al.* proposed in [81] a new definition of the low-Mach parameter :

$$\beta_v^2 = \max\left(\frac{\alpha_1\ (\alpha_1 - 1)}{\alpha_1 - 1 + c^2/u^2}\ ,\ \frac{\alpha_2\ (\alpha_2 - 1)}{\alpha_2 - 1 + c^2/v^2}\right) \tag{6.2.4}$$

with

$$\alpha_1 = \frac{1}{Re_x} = \frac{\rho^V}{u\delta x} \quad , \qquad \alpha_2 = \frac{1}{Re_y} = \frac{\rho^V}{v\delta y} \quad . \tag{6.2.5}$$

For the case that we considered, simple calculations yields :

$$\beta_v \approx \frac{\rho^V/\delta x}{c}$$

so that the eigenvalue is now equal to $\lambda_1^+ \approx \rho^V/\delta x$ which leads to :

$$CFL_x = \frac{\lambda_1^+ \cdot \Delta t}{\delta x} \approx \frac{\rho^V/\delta x \cdot \Delta t}{\delta x} = VNN_x$$

Consequently, the viscous preconditioning allows to optimize the VNN number and the CFL number at the same time. The definition of the low-Mach parameter for the viscous case is therefore the following :

$$\beta^2 = \min[\max[M^2, \beta_v^2], 1]$$

We can notice that there is no need of additional data in order to define the low-Mach parameter.

6.3 Impact of viscous spectral radius simplification for subsonic cases

In order to examine precisely the behaviour of the Matrix-Free scheme for the Navier-Stokes equations, we distinguish four types of implicit stages. The first one is the classical Block scheme for which inviscid and viscous terms are evaluated using block Jacobians. The second one is an

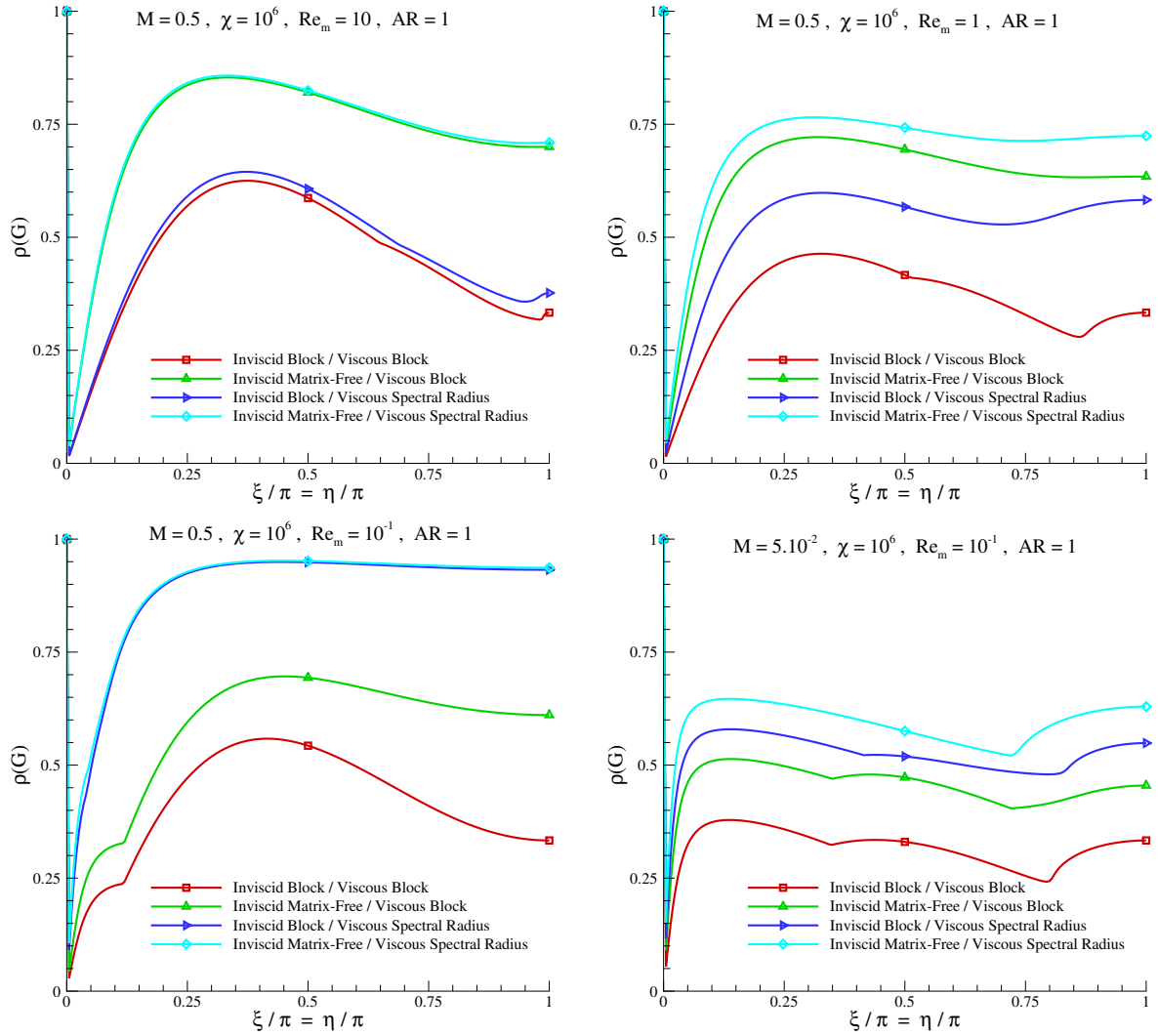


Figure 6.3.1: Amplification factor profiles along diagonal axis for the four different implicit stages. Top-Left : $M = 0.5$, $Re_m = 10$. Top-Right : $M = 0.5$, $Re_m = 1$. Bottom-Left : $M = 0.5$, $Re_m = 0.1$. Bottom-Right : $M = 0.05$, $Re_m = 0.1$

hybrid stage for which the inviscid terms are simplified using spectral radius while the viscous terms are treated like the Block scheme. The third stage is built using inviscid Jacobians and viscous spectral radius. Then, the last one is our Matrix-Free scheme for which both inviscid and viscous terms are evaluated thanks to spectral radius. In practice, the second and the third schemes are not really interesting because they are expected to provide less efficiency without reducing the memory storage. Nonetheless, using theses four implicit stages enables us to know which simplification, inviscid or viscous one, is mainly penalizing to obtain good efficiency. In this section, we consider subsonic flows and we vary the cell Reynolds number from convective dominated flows ($Re_m > 1$) to viscous dominated flows ($Re_m < 1$). The results in figure 6.3.1 present the amplification factor along the diagonal axis of the wavenumber domain. We first set the Mach number at $M = 0.5$ and we took successively the cell Reynolds number at 10, 1 and 0.1. We observe that for $Re_m = 10$, the loss of efficiency is mainly due to the inviscid spectral

radius simplification since the simplification in the viscous part does not deteriorate efficiency. On the other hand, at $Re_m = 0.1$ the viscous spectral radius is the cause of the bad damping properties. For $Re_m = 1$, that is when convective and viscous effects are of the same order, the loss of efficiency is equally distributed between the two simplification. According to this first analysis, one can think that the effects of the viscous spectral simplification on the efficiency are mainly driven by the value of the cell Reynolds number, as this latter decreases the viscous simplification becomes more penalizing. In fact, this is a bit more complex, indeed, by keeping the cell Reynolds number at 0.1 and setting the Mach number at $M = 0.05$ we can notice that the effects of both simplification become of the same order. Consequently, the impact of the viscous spectral radius simplification seems to rely on the ratio Re_m/M rather than Re_m. This ratio is known as the "acoustic" Reynolds number and it is denoted as follows :

$$Re_c = \frac{c}{\nu/\delta x} \tag{6.3.1}$$

When $Re_c > 1$, acoustic effects dominate the flow, thus the viscous spectral radius simplification does not affect the efficiency of the method. On the other hand, when $Re_c < 1$, the viscous characteristic time is the smaller and the simplification on the viscous terms is therefore highly penalizing. However, in our practical computations, the "acoustic" Reynolds number is rarely smaller than unity. We can reasonably think that the viscous spectral radius does not cause too big a loss of efficiency. In the rest of the chapter, our studies will aim at low-speed, viscous-dominated flows, that is for $Re_m << 1$, $M << 1$ and $Re_c > 1$.

6.4 Low-Mach viscous-dominated regime

In this section, we want to study precisely the efficiency of the Matrix-Free method in the case of low-Mach viscous dominated flows. Before tackling this subject, let us first study the impact of the viscous preconditioning on the implicit block scheme. We set the Mach number at $M = 10^{-5}$ and the cell Reynolds number at $Re_m = 10^{-1}$ and we draw the amplification factor map for the inviscid and the viscous preconditioning (*cf.* figure 6.4.1). The viscous preconditioning allows to increase dramatically the intrinsic efficiency of the Block scheme over the entire wavenumber domain. As mentioned earlier, it approximately equalizes the acoustic-CFL number and the VNN number which removes the stiffness from the Navier-Stokes system.

We now make the same study for the Matrix-Free scheme. The results are presented in figure 6.4.2. Unlike the Block case, the viscous preconditioning does not improve significantly the intrinsic efficiency of the Matrix-Free method. We even note that the inviscid preconditioning provides a better damping for the low-frequency modes. The reasons of such a behaviour are not clearly indentified, we suppose that the viscous spectral radius simplification is at the origin of the problem. Nevertheless, we tried to find a mean so as to improve the efficiency of the method. This research led us to modify the viscous preconditioning definition.

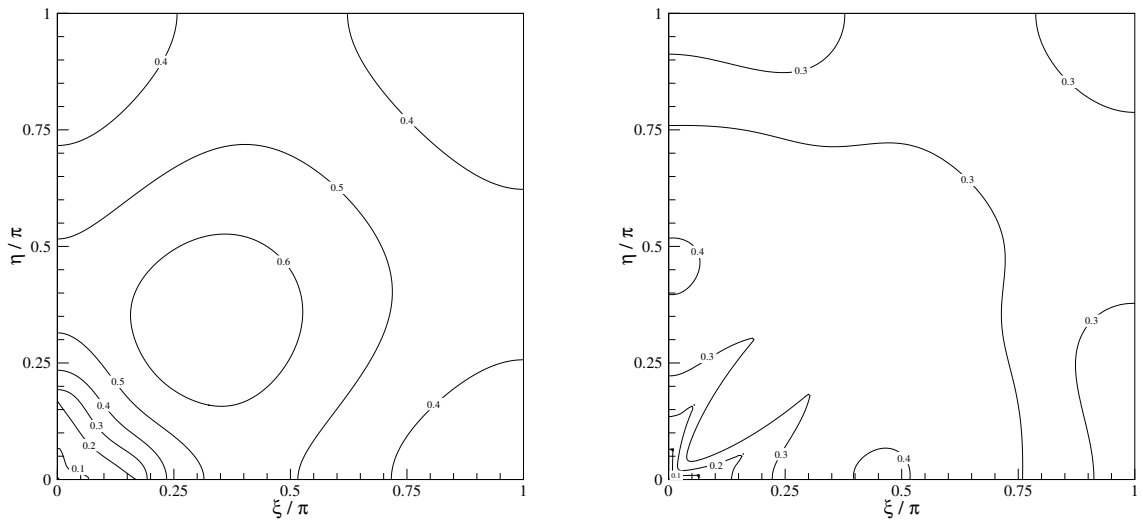


Figure 6.4.1: Amplification factor map of the Block scheme solved directly at $M = 10^{-5}$, $Re_m = 10^{-1}$ and $\chi = 10^6$. Left : Inviscid preconditioning. Right : Viscous preconditioning.

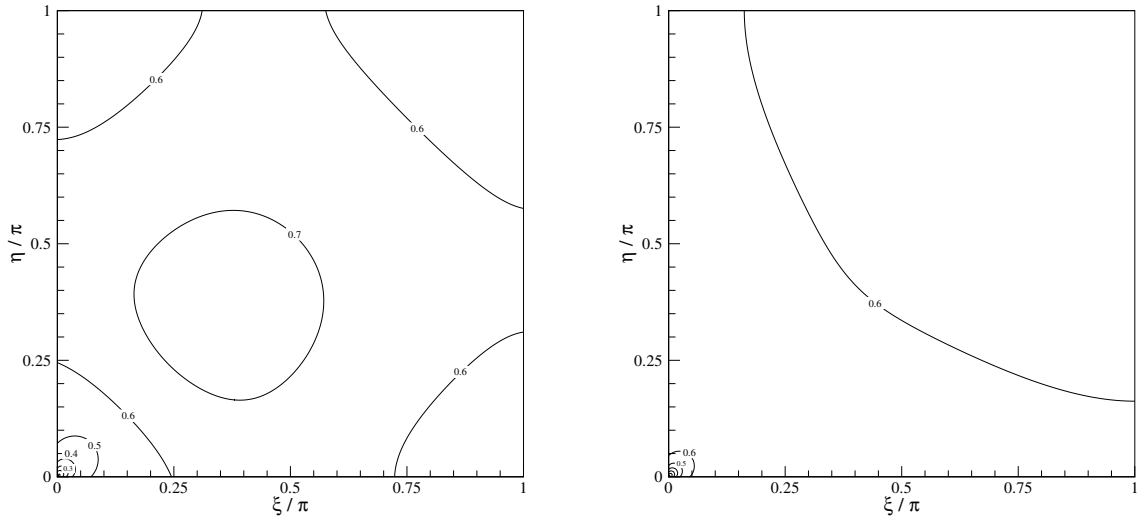


Figure 6.4.2: Amplification factor map of the Matrix-Free scheme solved directly at $M = 10^{-5}$, $Re_m = 10^{-1}$ and $\chi = 10^6$. Left : Inviscid preconditioning. Right : Viscous preconditioning.

6.5 Viscous preconditioning enhancement

First researches to increase the damping of the Matrix-Free scheme for the low-Mach viscous dominated regime revealed that the optimum value of the low-Mach parameter β settled between β_v and M. For the cases that we studied, we usually have $\beta_v > M$ which means that the viscous preconditioning creates more numerical dissipation than the inviscid one. As the viscous spectral radius simplification provides over-dissipation, we tried to reduce this numerical dissipation by setting :

$$\beta_{MF}^2 = \beta_v \times M$$

The results obtained with this choice are presented for the Block and the Matrix-Free schemes in figure 6.5.1. We notice that this new version of the viscous preconditioning does not enable the Block scheme to be more efficient compare to the classical version. On the other hand, the Matrix-Free scheme reaps a great benefit of this new choice. Indeed, there is a significant improvement of the damping over the entire wavenumber domain and the performances in the very low-frequency region are now very similar to the inviscid preconditioning. Thus, the classical version of the viscous preconditioning is clearly well-suited for the Block scheme while the Matrix-Free algorithm requires an alternative version. In the rest of the paragraph we will investigate further the impact of this enhanced preconditioning on the Matrix-Free method.

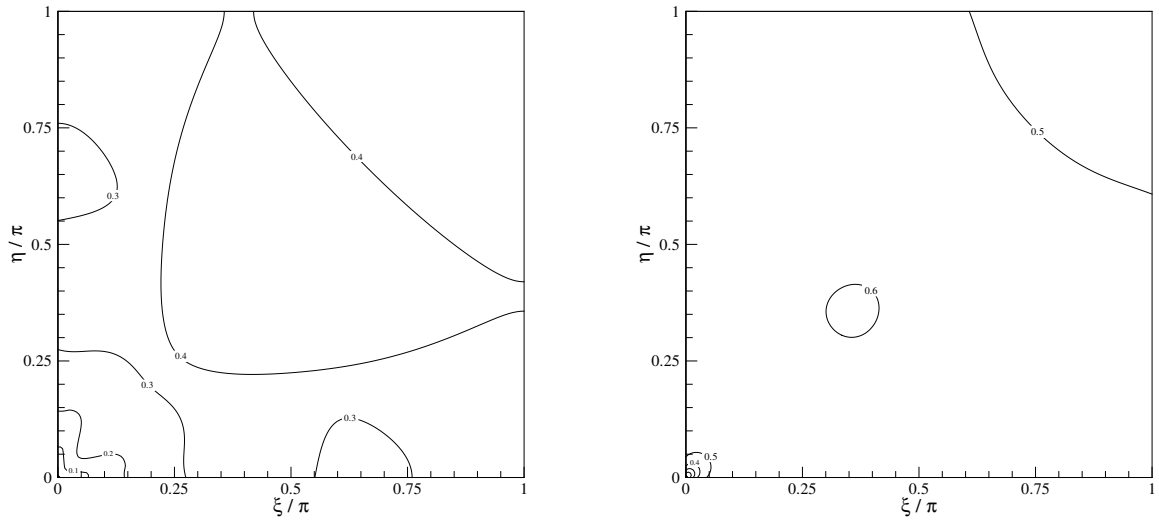


Figure 6.5.1: Amplification factor map of the Direct Solver with the enhanced preconditioning at $M = 10^{-5}$, $Re_m = 10^{-1}$ and $\chi = 10^6$. Left : Block scheme. Right : Matrix-Free scheme.

The good properties of the new definition of the viscous preconditioning remain valid provided that the acoustic Reynolds number is greater than unity. So as to confirm this interesting behaviour, we present in figures 6.5.2 and 6.5.3 some comparisons between all the types of preconditioning in the case of the Matrix-Free scheme solved directly. We draw the amplification factor along the diagonal line $\xi = \eta$. For every case, the inviscid preconditioning provides a better damping of the low-frequency modes while the viscous preconditioning is more efficient in the rest of the wavenumber domain. The new definition enables to preserve both types of

efficiency. Indeed, we note that the enhanced preconditioning provides a similar damping than the inviscid one for the low-frequency modes while it is more efficient than the viscous one in the rest of the wavenumber domain.

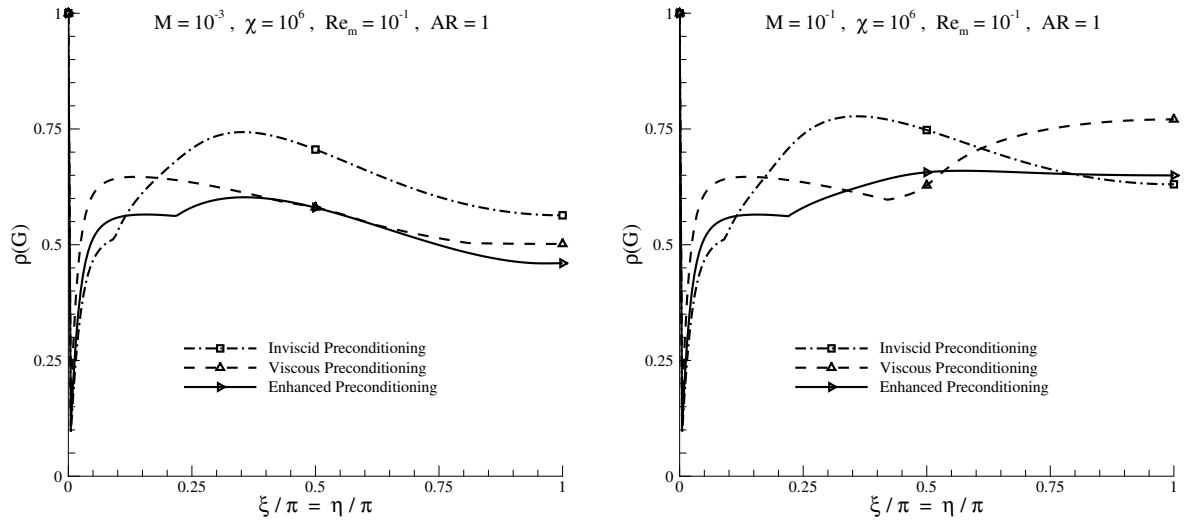


Figure 6.5.2: Amplification factor of the Matrix-Free scheme solved directly along the diagonal line $\xi = \eta$. Left : Comparison at $M = 10^{-3}$, $Re_m = 10^{-1}$ and $\chi = 10^6$. Right : Comparison at $M = 10^{-1}$, $Re_m = 10^{-1}$ and $\chi = 10^6$.

Thanks to this enhanced definition of the viscous preconditioning, the loss of efficiency due to the Matrix-Free simplifications is reduced. The method can be then very competitive with respect to the Block scheme. Next paragraph is devoted to the study of the Matrix-Free scheme solved by SGS or PJ relaxation procedures. Given to the good results obtained with the enhanced preconditioning, we will only consider it for the MF schemes.

6.6 MF-SGS scheme and MF-PJ scheme

MF-SGS and MF-PJ algorithms appeared to be competitive methods to solve Euler equations without high aspect ratio (*cf.* chapter 5). Indeed, both methods provide good damping for a reasonable number of inner-iterations. We expect to obtain the same properties for the Navier-Stokes equations. Let us consider the case detailed in the previous section ($M = 10^{-5}$ and $Re_m = 10^{-1}$). In figure 6.6.1, we plotted the amplification factors of the MF-SGS(40) method (on the left) and of the MF-PJ(50) algorithm (on the right).
As expected, MF-SGS scheme enables us to obtain damping properties close to the ones derived from the Direct scheme. However, we note that for the very low wavenumbers, MF-SGS scheme is a little bit stiffer than the Direct method, indicating a loss of efficiency. Despite this, MF-SGS scheme can be regarded as a valuable choice to solve the Navier-Stokes equations.
Similarly, MF-PJ(50) method provides a good damping above a large part of the wavenumber domain, nonetheless, it becomes really stiff for the low-frequency area. This issue can be partially circumvented by using more inner-iterations at the expense of an increase of the cost per non-linear iteration. However, MF-PJ scheme remains also competitive and we can notice that, unlike

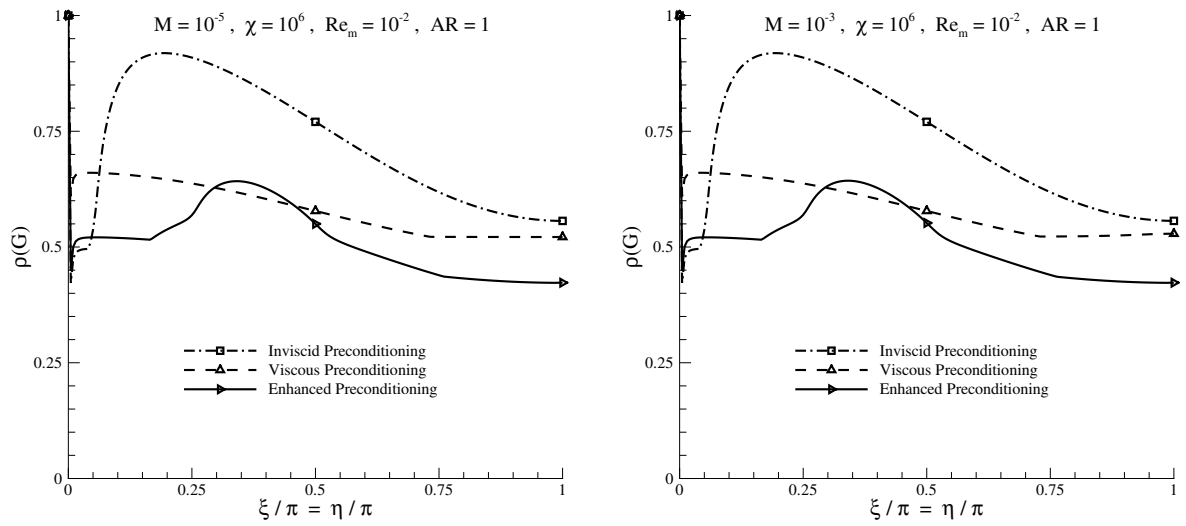


Figure 6.5.3: Amplification factor of the Matrix-Free scheme solved directly along the diagonal line $\xi = \eta$. Left : Comparison at $M = 10^{-5}$, $Re_m = 10^{-2}$ and $\chi = 10^6$. Right : Comparison at $M = 10^{-3}$, $Re_m = 10^{-2}$ and $\chi = 10^6$.

for the Euler equations, the method remains stable for high value of the safety factor χ.
In the previous chapter, we observed that both algorithm were not well-suited for solving Euler equations in the presence of high aspect ratio grids. Using stretched grids is customary to solve the Navier-Stokes equations since we need more points to describe the flow in the shear layer. It is therefore necessary to analyze the performance of SGS and PJ algorithms in such conditions. This is the objective of the next section.

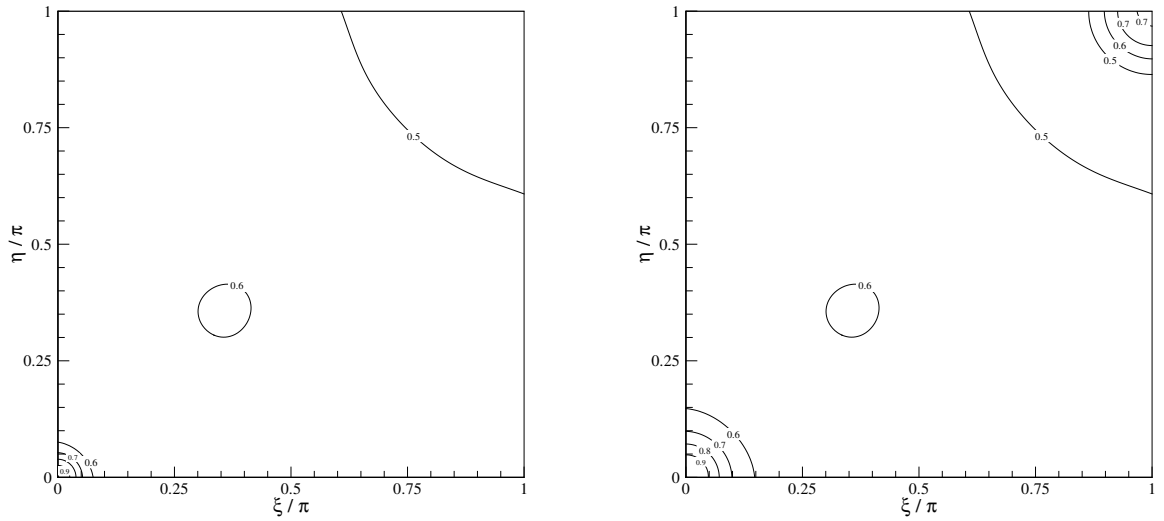


Figure 6.6.1: Amplification factor map at $M = 10^{-5}$, $Re_m = 10^{-1}$ and $\chi = 10^6$. Left : MF-SGS(40) scheme with enhanced viscous preconditioning. Right : MF-PJ(50) scheme with enhanced viscous preconditioning.

6.7 Effect of high aspect ratio

High grid aspect ratio are customary encountered in practical computations since shear layers require a large amount of points in the direction normal to the wall in order to resolve the high gradient regions. In chapter 5, we dealt with such high aspect ratio issues for the Euler equations and we showed that the Matrix-Free simplification did not generate any difficulties. On the other hand, we noticed that SGS and PJ algorithms were not well-suited for problems involving very stretched meshes. Theses methods are expected to behave similarly for the Navier-Stokes equations. In order to show the difficulty of dealing with viscous flows over high aspect ratio grids, let us consider an example. Let us suppose that $Re_m < 1$, $M << 1$ and $AR << 1$. We suppose also that we use the classical viscous preconditioning and we compute the CFL numbers and the VNN numbers in each direction as follows :

$$CFL_x = \frac{\lambda_1^+ \ \Delta t}{\delta x} \approx \frac{\rho^V \ \Delta t}{\delta x \delta y} = VNN_y \times AR$$

$$CFL_y = \frac{\lambda_2^+ \ \Delta t}{\delta y} \approx \frac{\rho^V \ \Delta t}{\delta y^2} = VNN_y$$

$$VNN_x = \frac{\rho^V \ \Delta t}{\delta x^2} = VNN_y \times AR^2$$

$$VNN_y = \frac{\rho^V \ \Delta t}{\delta y^2}$$

In the case of grids stretched in the x-direction, the viscous preconditioning enables to optimize the VNN and the CFL numbers in the cross-stream direction. As a result, the CFL_x number and the VNN_x number are maintained at small value if the aspect ratio is very small, indicating a poor damping of the longitudinal error modes. Such a strategy is refered as the "max-CFL/max-VNN" combination since it optimizes the highest value of CFL and VNN parameters. The modification of the viscous preconditioning for the Matrix-Free scheme does not change this situation. To circumvent this issue, Venkastewaran and Merkle proposed in [81] a modification of the viscous preconditioning. Their idea is to simultaneously optimize the viscous processes in the cross-stream direction with the acoustic wave propagation in the longitudinal direction. In our case, the new definition of the viscous preconditioning tries to maintain the CFL_x number and the VNN_y number at their optimum values. This can be achieved by changing the definition (6.2.5) of α_1 and α_2 as follows :

$$\alpha_1 = \frac{1}{Re_x} \cdot \frac{1}{AR^2} \quad , \qquad \alpha_2 = \frac{1}{Re_y} \cdot AR^2 \quad . \tag{6.7.1}$$

This strategy is refered as the "min-CFL/max-VNN" combination since it optimises the minimum CFL number and the maximum VNN number. However, even with this new preconditioning the VNN_x number remains very small which still creates stiffness for the pure longitudinal modes. Moreover, this definition of the preconditioning provides more numerical dissipation. Indeed, simple calculations yields :

$$\beta_v \approx \frac{\rho^V / \delta y}{c} \cdot \frac{1}{AR} >> \frac{\rho^V / \delta y}{c}$$

When the aspect ratio is very small, the preconditioned scheme can even turn back to its non-preconditioned version. Hence, such a strategy seems clearly unadapted to our Matrix-Free scheme which requires less numerical dissipation, as mentioned earlier. The analysis that we performed indicates clearly that, for this choice, the Matrix-Free scheme is stiff over the entire wavenumber domain. A last strategy was also designed by Merkles' team [11, 81] and refered as the "min-CFL/min-VNN" combination since it tries to maintain the minimum CFL and VNN numbers at their optimal value. This choice can be enforced by taking the minimum instead of the maximum in expression (6.2.4) :

$$\beta_v^2 = \min\left(\frac{\alpha_1\ (\alpha_1 - 1)}{\alpha_1 - 1 + c^2/u^2}\ ,\ \frac{\alpha_2\ (\alpha_2 - 1)}{\alpha_2 - 1 + c^2/v^2}\right)$$

with the original definition of the coefficients α_1 and α_2. This choice enables to obtain a very good damping in the longitudinal direction, however, it implies that the values of the CFL number and the VNN number in the cross-stream direction are very big which can generate some stiffness over the frequency domain. Moreover, the choice of the minimum in the viscous preconditioning implies that the low-Mach parameter β can become equal to the Mach number as for the inviscid preconditioning and resulting stiffness can then appear. Next paragraph is devoted to the study of these different strategies in the case of the Direct Scheme.

6.7.1 Direct scheme

Let us consider a flow at $M = 10^{-5}$, $Re_m = 10^{-1}$ and let us take $AR = 10^{-2}$. We plotted in figure 6.7.1 the results obtained for the Block scheme at $\chi = 10^6$ and with the "max-CFL/max-VNN" preconditioning. The scheme provides an excellent damping over the entire frequency domain and even for the pure-longitudinal modes as it can be seen on the close-up. These good properties along the x-axis are mainly due to the use of high values of time-step (we recall that $\chi = 10^6$). Hence, in that case, we can estimate that $CFL_x \approx 10^4$ while $VNN_x \approx 10^2$ which are still high values for these parameters.
In figure 6.7.2, we present the results obtained with the Matrix-Free scheme for the same conditions except that we used the modified viscous preconditioning. We notice that the damping is good over a large part of the frequency domain. Nonetheless, we also note that the stiffness is mainly concentrated along the x-axis, as one can see in the close-up. Unlike the Block scheme, the damping of the Matrix-Free method deteriorates for the pure longitudinal modes and particularly for the high frequencies. Hence, it can be interesting to study the "min-CFL/min-VNN" combination and its effects on the performance rate for the pure longitudinal modes.
In figure 6.7.3, we plotted the amplification factor map obtained with the "min-CFL/min-VNN" strategy. As expected, the damping over the wavenumber domain is worse than with the previous version. High values of the transverse CFL and VNN numbers and the fact that the viscous preconditioning equalizes the inviscid case are the main reasons of the stiffness. On the other hand, as one can see in the close-up, the damping of the pure longitudinal modes is improved, particularly for the high frequencies and the very small wavenumbers. However, Venkateswaran and Merkle emphasize that with this kind of strategy the solutions may diverge and they propose to use the more conservative viscous preconditioning in the begining of the computation and then switch to the "min-CFL/min-VNN" combination. At any case, the efficiency for pure axial modes of the Matrix-Free scheme will rely also on the resolution method. Next section will precisely deal with the performances of the SGS and PJ relaxation procedure with high aspect ratio grids.

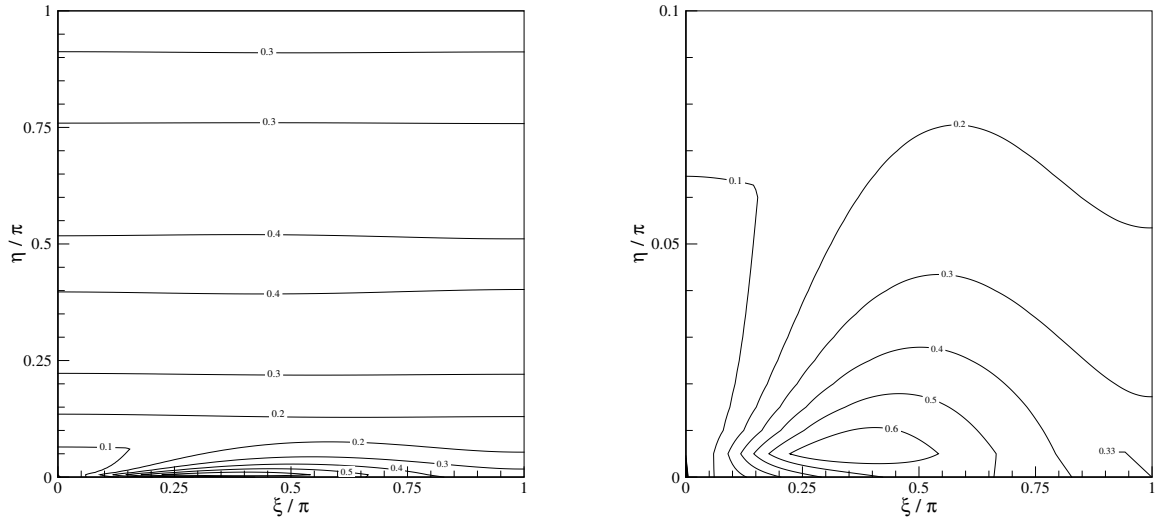


Figure 6.7.1: Amplification factor map of the Direct Block scheme at $M = 10^{-5}$, $Re_m = 10^{-1}$, $\chi = 10^6$ and $AR = 10^{-2}$. Left : Entire wavenumber domain. Right : Close-up near the absciss axis.

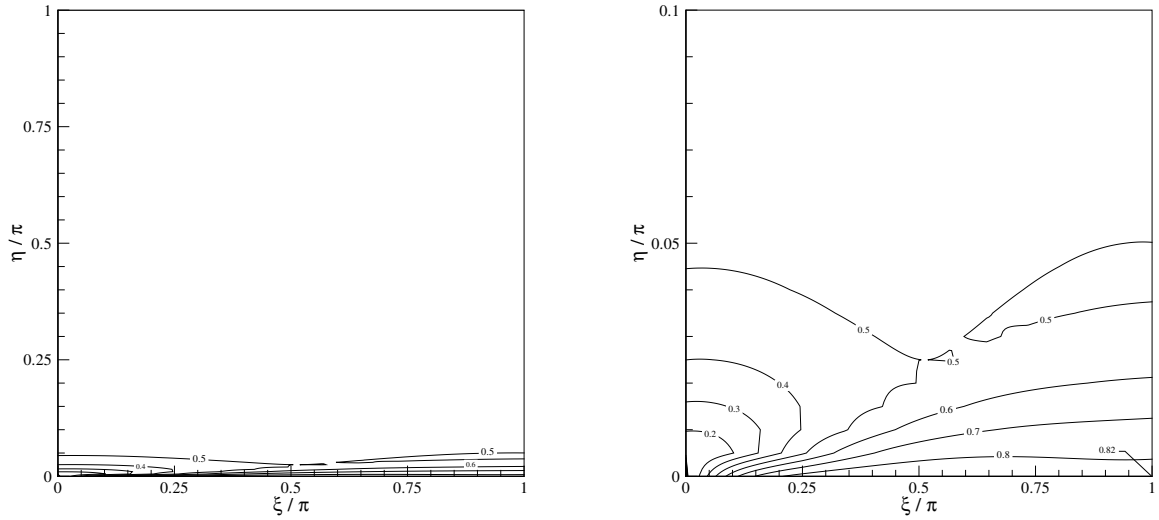


Figure 6.7.2: Amplification factor map of the Direct Matrix-Free scheme at $M = 10^{-5}$, $Re_m = 10^{-1}$, $\chi = 10^6$ and $AR = 10^{-2}$. Left : Entire wavenumber domain. Right : Close-up near the absciss axis.

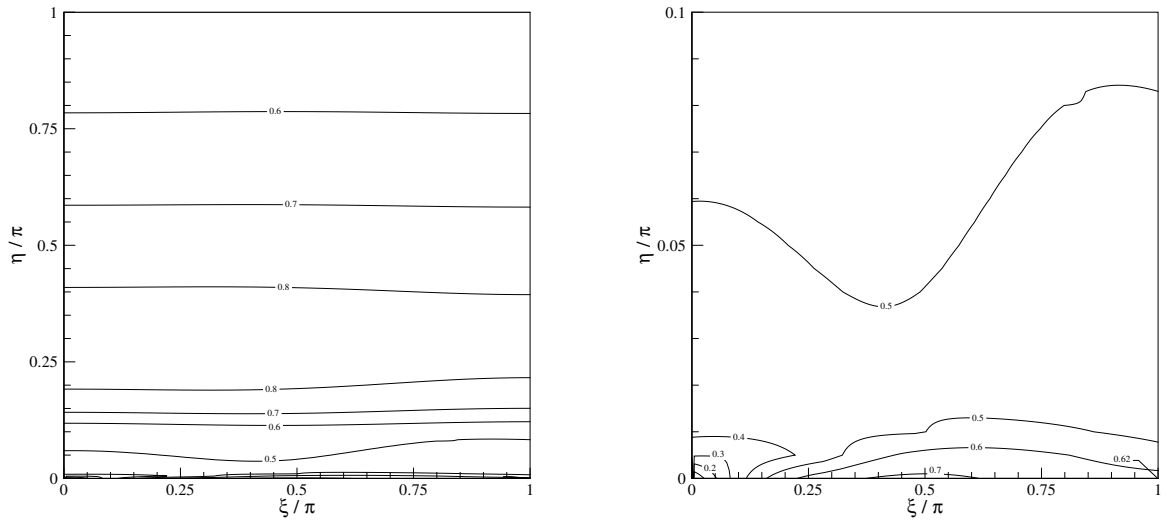


Figure 6.7.3: Amplification factor map of the Direct Matrix-Free scheme at $M = 10^{-5}$, $Re_m = 10^{-1}$, $\chi = 10^6$ and $AR = 10^{-2}$. The "min-CFL/min-VNN" strategy is used. Left : Entire wavenumber domain. Right : Close-up near the absciss axis.

6.7.2 MF-SGS and MF-PJ schemes

In this paragraph, we aim at studying the performances of MF-SGS and MF-PJ schemes in the presence of high aspect ratio grids. As mentioned earlier, they are expected to provide poor damping for the pure axial modes. Results from figure 6.7.4 show that for $AR = 10^{-2}$, both methods are efficient to damp the mid-wavenumber modes but they fail to eliminate pure longitudinal errors. Using "min-CFL/min-VNN" definition does not enable to improve significantly the performances. In figures 6.7.5 and 6.7.6, we present the amplification factor along the x-axis for the MF-SGS(40) and MF-PJ(50) schemes. For both schemes, the "min-CFL/min-VNN" strategy fails to improve the pure axial damping. Further investigations revealed that taking a very large amount of inner-sweeps enables both method to recover a little intrinsic efficiency but such a strategy is not affordable in terms of CPU-time efficiency. As a result, the use of the "min-CFL/min-VNN" combination cannot be reasonably considered for the Matrix-Free method coupled with SGS or PJ algorithm.

6.8 Conclusions

In this chapter, we studied the impact of the viscous spectral radius and we emphasized that, as long as $Re_c > 1$, such a simplification does not affect the efficiency of the method. When $Re_c < 1$, the simplification is highly penalizing but we can reasonably consider that such a case is rare for low-Mach number flows. Besides, we showed that the original viscous preconditioning is not really well-suited for the Matrix-Free technique since it provides too much numerical dissipation. Hence, we outlined a new definition of the viscous parameter which enables the method to provide a good damping for every frequency modes. In the case of high aspect ratio

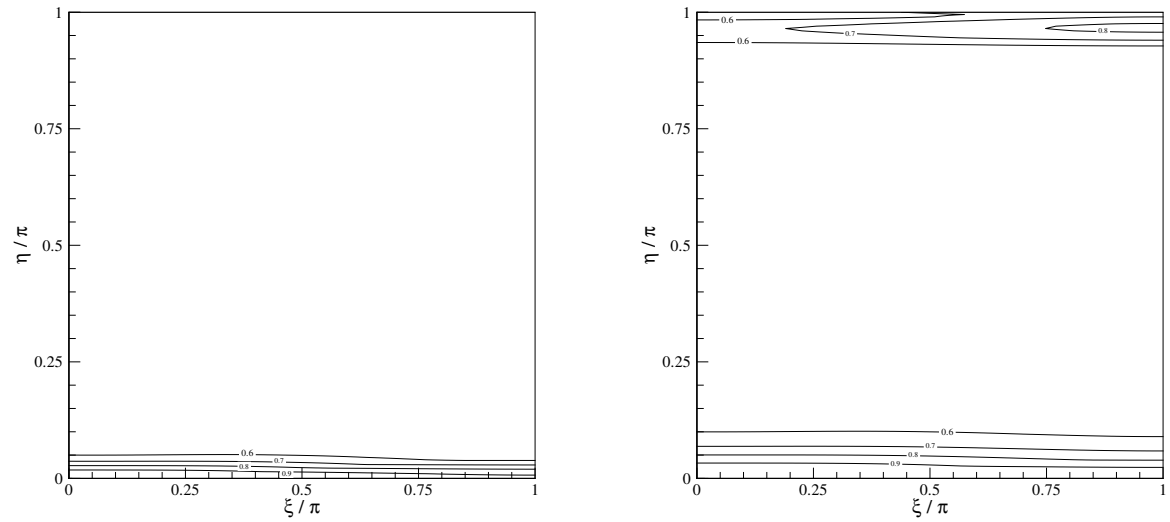


Figure 6.7.4: Amplification factor map at $M = 10^{-5}$, $Re_m = 10^{-1}$, $\chi = 10^6$ and $AR = 10^{-2}$. Left : MF-SGS(40) scheme with enhanced viscous preconditioning. Right : MF-PJ(50) scheme with enhanced viscous preconditioning.

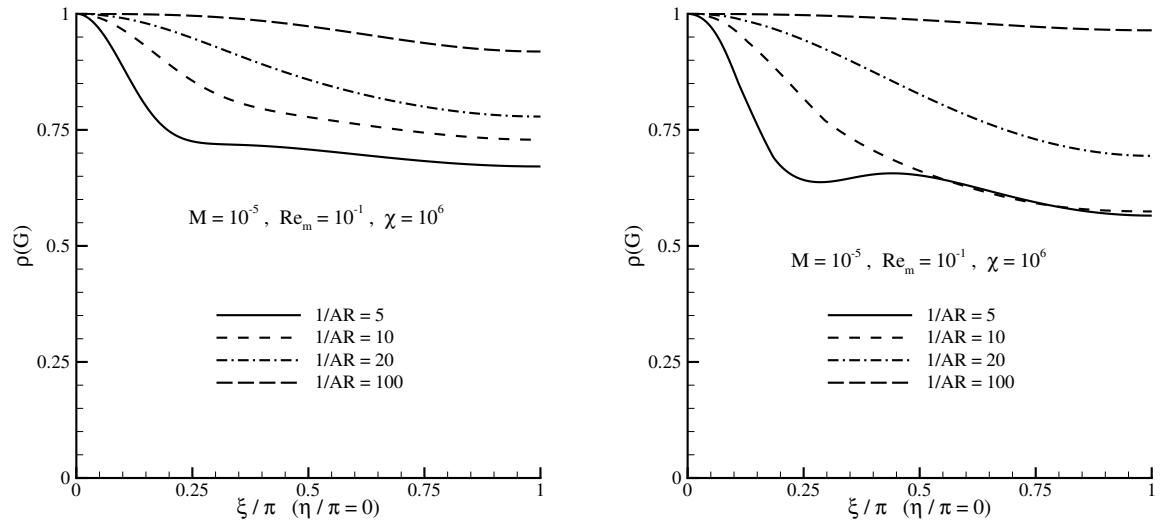


Figure 6.7.5: Amplification factor of the MF-SGS(40) scheme along the x-axis. Left : Enhanced viscous preconditioning. Right : "min-CFL/min-VNN" combination.

grids, we showed that the "min-CFL/min-VNN" combination can increase the efficiency of the method for the pure axial modes. Unfortunately, the same strategy fails to make SGS and PJ algorithms as efficient as the Direct scheme. However, some strategies are imaginable in order to avoid high aspect ratio issues, such as unstretching grids far from the wall boundaries so as to preserve efficiency. It was shown in [81] that axial modes can be eliminated by the boundary condition specifications. In chapter 8, we will focus on such problems and we will try to outline practical solutions.

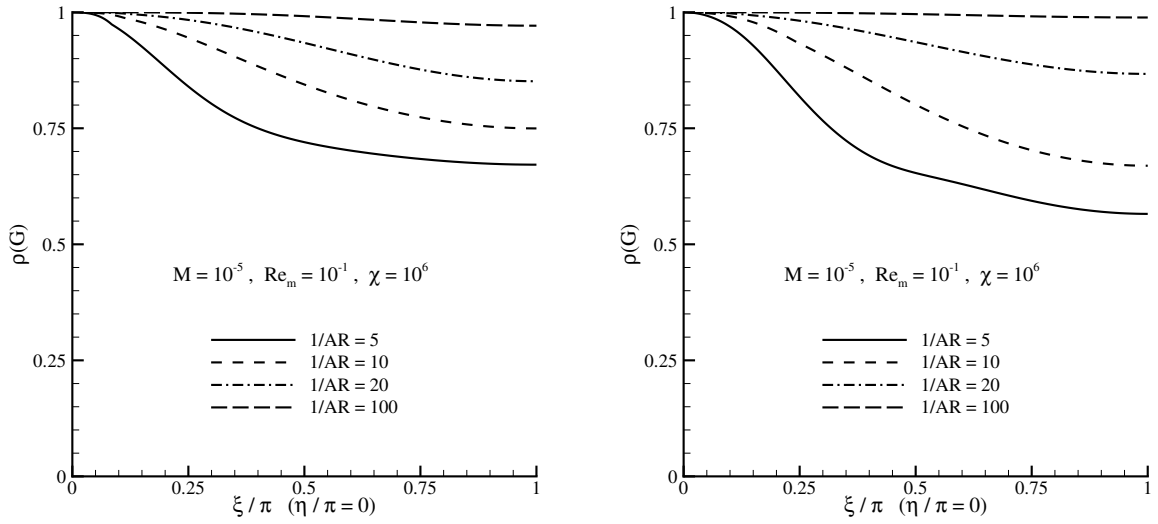


Figure 6.7.6: Amplification factor of the MF-PJ(50) scheme along the x-axis. Left : Enhanced viscous preconditioning. Right : "min-CFL/min-VNN" combination.

Chapter 7

Stability Analysis of the Matrix-Free Method for the Unsteady Equations

7.1 Introduction

Unsteady fluid dynamic processes are frequently encountered in engineering problems. As they are governed by a wide range of time scales (typically from acoustic time to particle time), they are challenging to compute. For low-Mach number flows, unsteady problem is even more complex since there is a strong disparity between the characteristic time related to acoustic waves and the characteristic time related to particle velocities. In this chapter, we focus on the Euler unsteady equations solved by a dual-time stepping scheme, wherein the physical-time derivatives are employed to follow the physical transients and the pseudo-time derivatives serve as an iterative device. In chapter 3, we showed how to obtain the expression of the dual-time stepping scheme in the Fourier space. For the Euler case, it is given by the following system :

$$H \cdot \Delta\widehat{w}_{i,j}^{n,m} = -K \cdot \widehat{w}_{i,j}^{n,m} \quad ,$$

with

$$\left\{ \begin{array}{rcl} K & = & \dfrac{3}{2}\dfrac{\Delta\tau}{\Delta t} Id \; + \; 2\dfrac{\Delta\tau}{\delta x}(1-\varepsilon)\tilde{D}_1 \; z_1 \; + \; 2\dfrac{\Delta\tau}{\delta y}(1-\varepsilon)\tilde{D}_2 \; z_2 \\ & & + \; \dfrac{\Delta\tau}{\delta x}\epsilon \; (1-\kappa)\tilde{D}_1 \; z_1^2 \; + \; \dfrac{\Delta\tau}{\delta y}\epsilon \; (1-\kappa)\tilde{D}_2 \; z_2^2 \\ & & + \; \underline{\mathrm{i}} \; \left(\dfrac{\Delta\tau}{\delta x} A^E \; s_1 \; \left(1 + \dfrac{\epsilon}{2} \; (1-\kappa) \; z_1 \right) + \dfrac{\Delta\tau}{\delta y} B^E \; s_2 \; \left(1 + \dfrac{\epsilon}{2} \; (1-\kappa) \; z_2 \right) \right) \\ H & = & P^{-1} + \; \dfrac{3}{2}\dfrac{\Delta\tau}{\Delta t} Id \; + \; 2\dfrac{\Delta\tau}{\delta x}\tilde{D}_1 \; z_1 \; + \; 2\dfrac{\Delta\tau}{\delta y}\tilde{D}_2 \; z_2 \; + \; \underline{\mathrm{i}} \left(\dfrac{\Delta\tau}{\delta x} A^E s_1 + \dfrac{\Delta\tau}{\delta y} B^E s_2 \right) \end{array} \right.$$

where we recall that Δt is the physical time-step and $\Delta\tau$ is the pseudo-time step. In order to perform a Von Neumann analysis, we need to customize the unsteady system. That is the subject of the next paragraph.

7.2 Parameters of the study

In the manner of the previous chapters, we need parameters which enable us to compute the Jacobian matrices and the dissipation matrices. As shown earlier, the Mach number M, the flow direction v/u and the ratio of specific heats γ are sufficient to determine such matrices. Besides, we have to compute the ratios $\Delta\tau/\delta x$, $\Delta\tau/\delta y$ and $\Delta\tau/\Delta t$. The pseudo-time step reads :

$$\Delta\tau = CFL_\tau \cdot \min(\frac{\delta x}{\lambda_1^+}, \frac{\delta y}{\lambda_2^+})$$

where λ_1^+ and λ_2^+ are the spectral radius of the preconditioned system. Hence, we obtain :

$$\begin{cases} \dfrac{\Delta\tau}{\delta x} = CFL_\tau \cdot \min(\dfrac{1}{\lambda_1^+}, \dfrac{AR}{\lambda_2^+}) \\ \dfrac{\Delta\tau}{\delta y} = CFL_\tau \cdot \min(\dfrac{1/AR}{\lambda_1^+}, \dfrac{1}{\lambda_2^+}) \end{cases} \tag{7.2.1}$$

Consequently, two others parameters are required, namely the aspect ratio $AR = \delta y/\delta x$ and the dual CFL number CFL_τ. Furthermore, we have to define the physical-time step. There are two possible definitions, the first one based on the acoustic time scale, the second one based on the particle time scale :

$$\Delta t = CFL_t \cdot \delta x \cdot \min(\frac{1}{u+c}, \frac{AR}{v+c}) \qquad \text{or} \qquad \Delta t = CFL_t \cdot \delta x \cdot \min(\frac{1}{u}, \frac{AR}{v})$$

At any case, we have :

$$\frac{\Delta\tau}{\Delta t} = \frac{\Delta\tau}{\delta x} \cdot \frac{\delta x}{\Delta t}$$

Thus, we choose the physical CFL number CFL_t as the additional parameter to set the physical time step. Before performing analyses, we want to define two practical numbers, namely the CFL number based on the particle velocity and denoted CFL_u, and the CFL number based on the acoustic wave and denoted CFL_{u+c} :

$$CFL_u = u \cdot \frac{\Delta t}{\delta x} \qquad \text{and} \qquad CFL_u = (u+c) \cdot \frac{\Delta t}{\delta x}$$

When the Mach number is very small, the scheme has to deal with a strong disparity between these two numbers. For instance, if the Mach number is set at $M = 10^{-3}$ and if we are interesting in the acoustic phenomena, then we have $CFL_{u+c} \approx 1$ and $CFL_u \approx 10^{-3}$. Hence, we can encounter some problems because of these very different time-scale. Let us now deal with the low-Mach parameter which enables us to circumvent such issues.

7.3 Unsteady preconditioning

As mentioned earlier, the large disparity between acoustic CFL number and particle CFL number that we encounter for low-Mach number flow obliges us to redefine properly the low-Mach parameter. Several studies about this problem are available in the literature and Merkle's teams

provided an enhancement of steady preconditioning in [81]. Using standard steady preconditioning can be specially ineffective for acoustic problems. In figure 7.3.1, we plotted the amplification factor obtained with the MF-SGS(30) scheme with the steady preconditioning (left) and without any preconditioning (right). We notice that the preconditioned scheme becomes particularly stiff for the low-frequency area and, as a result, it is inappropriate for such problems. On the other hand, the standard scheme appears to be well-suited for this case and provides a very good damping over the entire wavenumber domain.

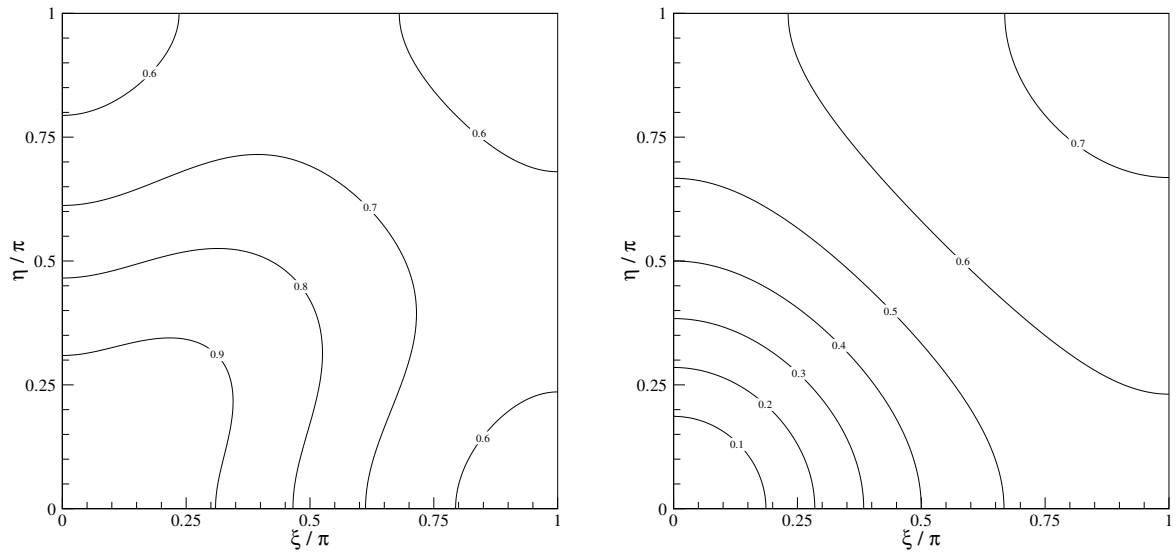


Figure 7.3.1: Amplification factor of the MF-SGS(30) scheme at $M = 10^{-3}$, $CFL_{u+c} = 1$ and $CFL_\tau = 10^6$. Left : Steady preconditioning. Right : No preconditioning.

For particle problems, that is when $CFL_u = 1$, the results are different. Indeed, in figure 7.3.2, we can note that the Steady preconditioning scheme provides a good damping while the unpreconditioned scheme is highly stiff for such a case. Steady preconditioning is then more appropriate for particle problems, however we remark that the damping over the low-frequency region is still ineffective, indicating slow convergence. Standard preconditioning is therefore not versatile enough to deal with the limits of small and large physical time-steps. Moreover, in the case of intermediate choices of time-step size, neither steady preconditioned scheme nor unpreconditioned one provide good convergence properties. Researches showed that an additional definition of the low-Mach parameter was then necessary. The unsteady parameter is defined as follows :

$$\beta_u = \max\left(\frac{l_x}{\pi\ \Delta t\ c}, \frac{l_y}{\pi\ \Delta t\ c}\right)$$

where l_x and l_y are characteristic lengths of the physical problem. In practice, it is customary to take these lengths equal to the size of the computational domain. The main reason is that we want to damp efficiently the low-frequency error modes since they usually control the convergence rate. These modes are related to the longest wavelengths that can be captured over the domain, it is therefore reasonable to consider that such wavelengths are of the order of the size of the computational domain. As a result, the ratio $l_x/\Delta t$ can be regarded as the speed of these waves and the unsteady parameter can be considered as the Mach number which defines them.

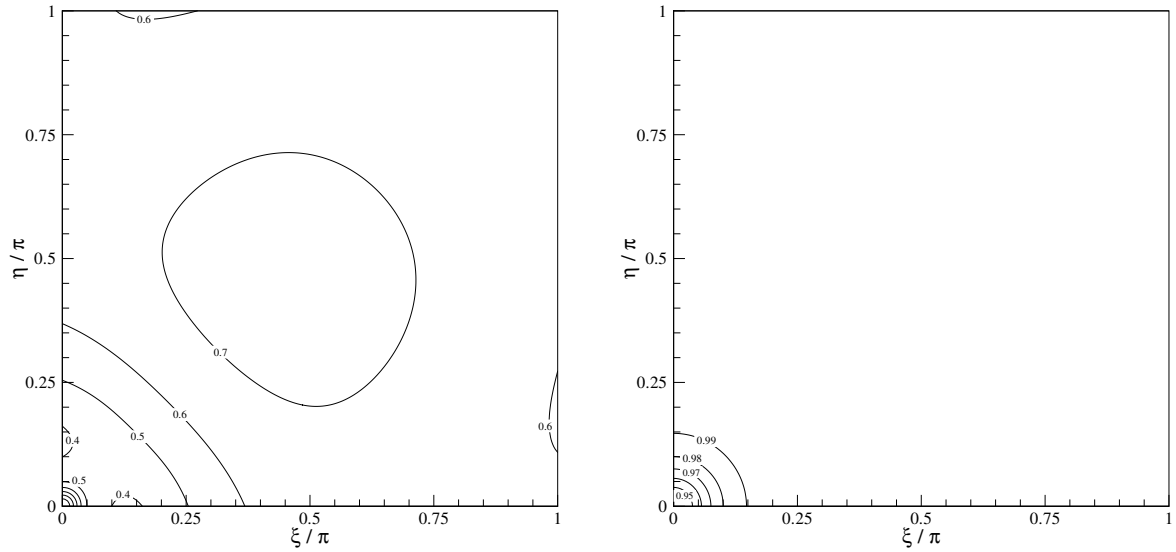


Figure 7.3.2: Amplification factor of the MF-SGS(30) scheme at $M = 10^{-3}$, $CFL_u = 1$ and $CFL_\tau = 10^6$. Left : Steady preconditioning. Right : No preconditioning.

However, there exists another way to determine these lengths. In fact, the convergence rate is controlled by the low-frequency modes but their representation in the Fourier space relies on the mesh size. More precisely, when the grid becomes finer, the representation of the largest wavelength corresponds to lower frequencies. Hence, there are two characteristic lengths which have to be taken into account, namely the size of the wave and the size of the mesh. Considering the fastest wave propagating at the speed $(u + c)$ during the physical time-step Δt, we obtain the following length :

$$L_{acous} = (u + c) \cdot \Delta t = CFL_{u+c} \cdot \delta x$$

According to the value of CFL_{u+c}, this length can be seen as the number of space-step that the fastest wave covers during one time-step. Moreover we can consider that it represents the scale of the largest wavelength. As the frequency representation depends also on the mesh size, we can take as reference length the geometry average of $(L_{acous} \times \delta x)$:

$$l_x = \sqrt{L_{acous} \cdot \delta x} = \delta x \cdot \sqrt{CFL_{u+c}}$$

Thus, the unsteady parameter in the x-direction can be redefined as follows :

$$\beta_u = \frac{\delta x / \Delta t}{c} \cdot \frac{\sqrt{CFL_{u+c}}}{\pi}$$

Let us consider an acoustic problem, CFL_{u+c} is set to unity and the physical time-step is therefore equal to $\Delta t = \delta x/(u + c)$. The unsteady parameter is then given by :

$$\beta_u = \frac{u + c}{c} \cdot \frac{1}{\pi} \approx \frac{1}{\pi}$$

Using this parameter enables us to become very close to the unpreconditioned scheme which is well-suited for acoustic problems. On the other hand, let us set CFL_u to unity which means

that $\Delta t = \delta x/u$ and $CFL_{u+c} \approx 1/M$, we obtain :

$$\beta_u = \frac{u}{c} \cdot \frac{\sqrt{1/M}}{\pi} \approx \frac{\sqrt{M}}{\pi}$$

Now the unsteady parameter is close to the steady one and we can expect a good damping of the error modes. For intermediate cases, the square root of the acoustic CFL number enables us to scale properly the unsteady parameter so as to preserve good convergence properties. In figure 7.3.3, we show the results obtained with this definition of the unsteady parameter for $CFL_{u+c} = 1$ on the left and $CFL_u = 1$ on the right. We notice that for both cases, the scheme succeed in damping efficiently the low-frequency error modes, indicating fast convergence rate.

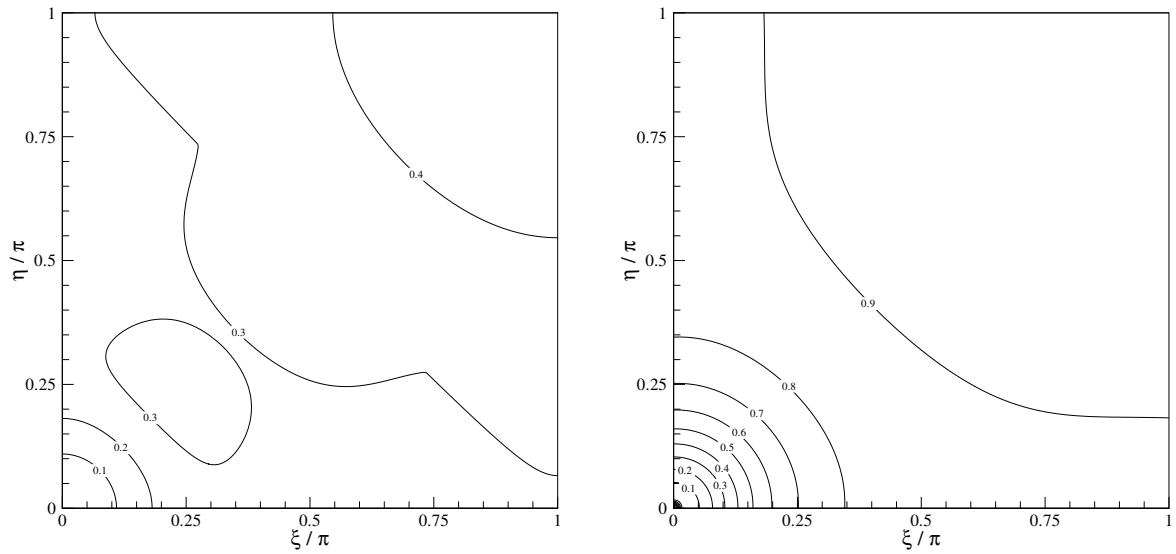


Figure 7.3.3: Amplification factor of the MF-SGS(30) scheme with the unsteady preconditioning at $M = 10^{-3}$ and $CFL_\tau = 10^6$. Left : $CFL_{u+c} = 1$. Right : $CFL_u = 1$.

In figure 7.3.4, we drew the diagonal profile of the amplification factor at $CFL_{u+c} = 1$ and $CFL_u = 1$. The unsteady preconditioner appears to be the more versatile method to damp correctly the low wavenumbers in both cases. Moreover, for intermediate configurations, namely for $CFL_u = 10^{-1}$ and for $CFL_u = 10^{-2}$, the unsteady preconditioner provides good convergence properties as shown in figure 7.3.5.

Remark :

We want to stress on the fact that the unsteady preconditioning has a non-negligible effect on the accuracy of the solution. Indeed, it implies that the low-Mach parameter is larger than for the steady preconditioner so that the numerical dissipation becomes more important. It is especially true in the case of the acoustic problems for which the unsteady preconditioning approaches the no preconditioning formulation. The numerical dissipation is then poorly scale leading to inaccuracy.

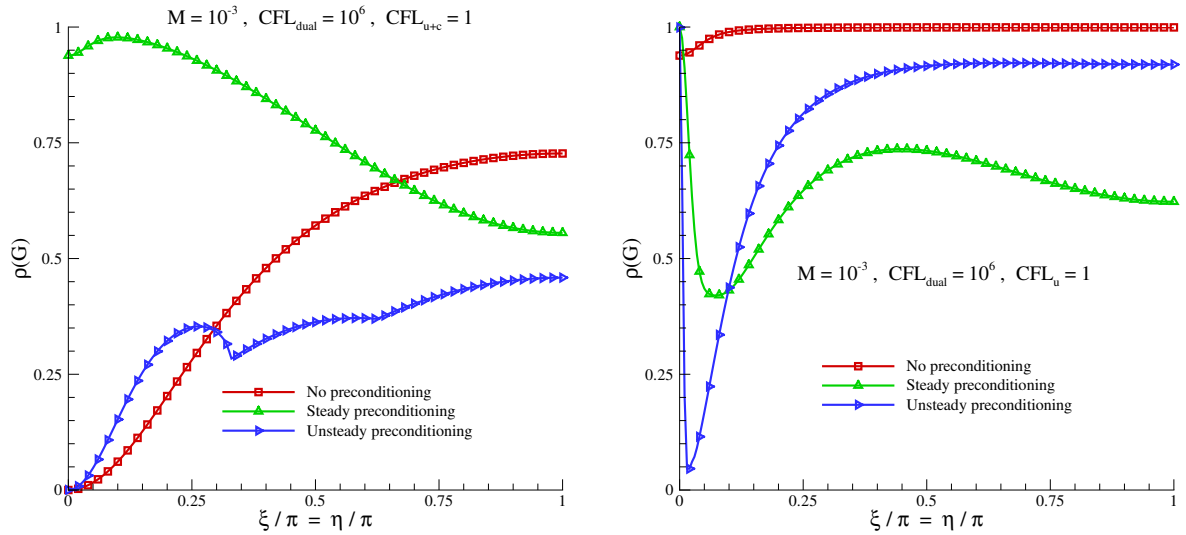


Figure 7.3.4: Diagonal profile of the amplification factor of the MF-SGS(30) scheme at $M = 10^{-3}$ and $CFL_\tau = 10^6$. Left : $CFL_{u+c} = 1$. Right : $CFL_u = 1$.

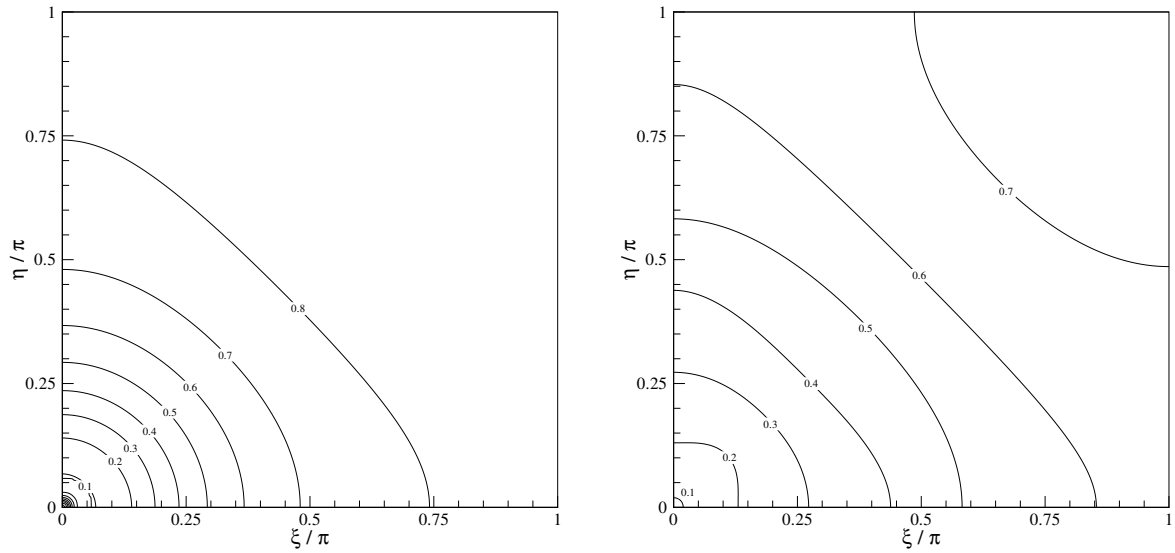


Figure 7.3.5: Amplification factor of the MF-SGS(30) scheme with unsteady preconditioning at $M = 10^{-3}$ and $CFL_\tau = 10^6$. Left : $CFL_u = 10^{-1}$. Right : $CFL_u = 10^{-2}$.

7.4 Comparison Block / Matrix-Free

7.4.1 Regular grids

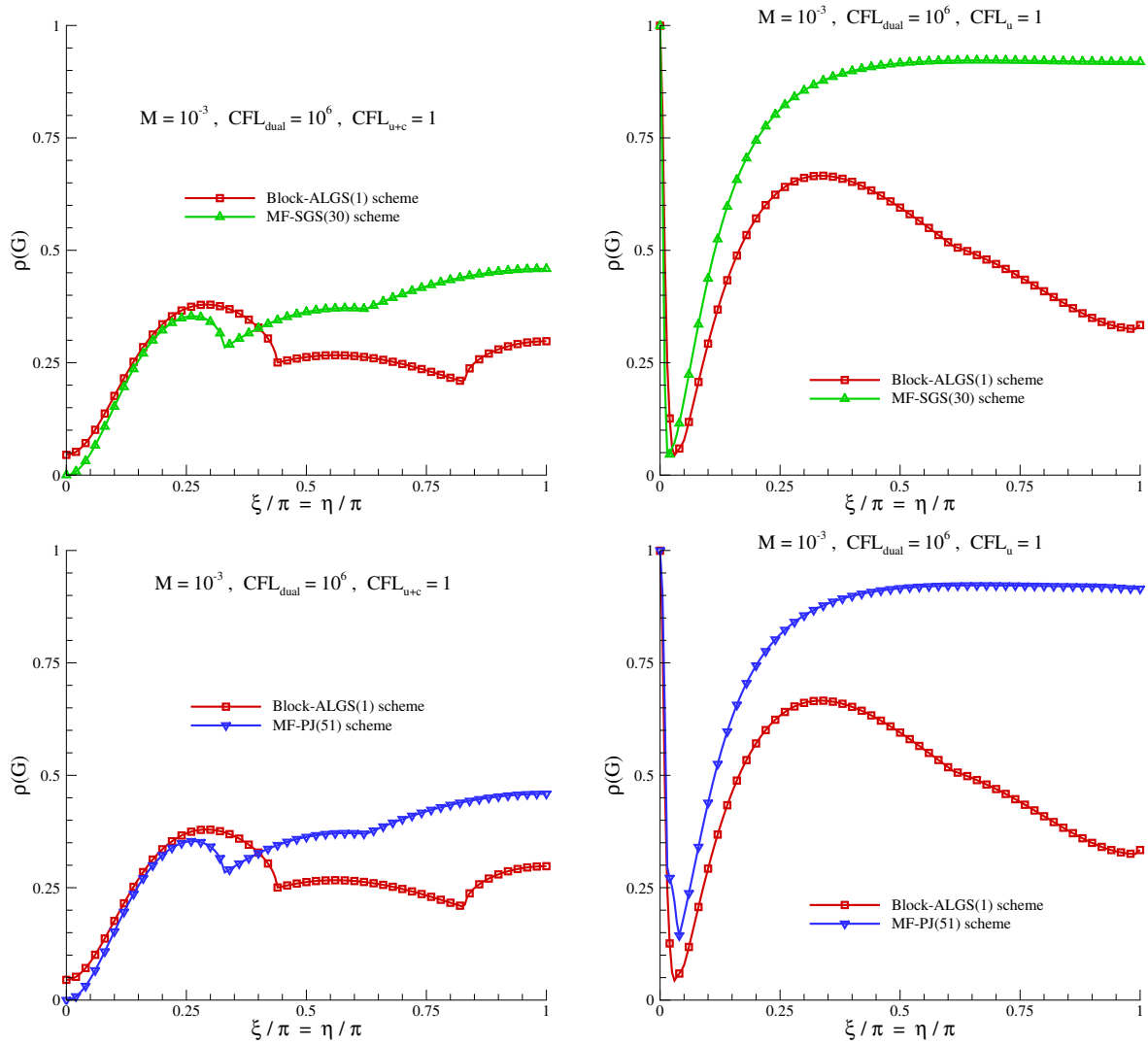


Figure 7.4.1: Diagonal profiles of the amplification factor at $M = 10^{-3}$ and $CFL_\tau = 10^6$ with the unsteady preconditioning. Top : Comparison between Block-ALGS(1) scheme and MF-SGS(30) scheme. Left : $CFL_{u+c} = 1$. Right : $CFL_u = 1$. Bottom : Comparison between Block-ALGS(1) scheme and MF-PJ(50) scheme. Left : $CFL_{u+c} = 1$. Right : $CFL_u = 1$.

The results presented in figure 7.4.1 are very interesting since they show that the Matrix-Free method coupled with SGS algorithm or PJ algorithm is as efficient as the Block scheme solved by ALGS procedure. Indeed, for acoustic and particle problems, that is for $CFL_{u+c} = 1$ or $CFL_u = 1$, the Matrix-Free scheme provides a very good damping of the low-frequency error modes which indicates a fast convergence rate. Thus, Matrix-Free method appears to be very appropriate for the unsteady flows.

7.4.2 Stretched grids

We first deal with the acoustic problems and we set the aspect ratio at 10^{-4}. For such a case, the MF-SGS scheme is very competitive since it provides an excellent damping over the entire wavenumber domain and it eliminates completely the pure axial modes (*cf.* figure 7.4.2). These results are surprising because we expected that the method provided a bad damping of the longitudinal modes as it is the case for steady flows.

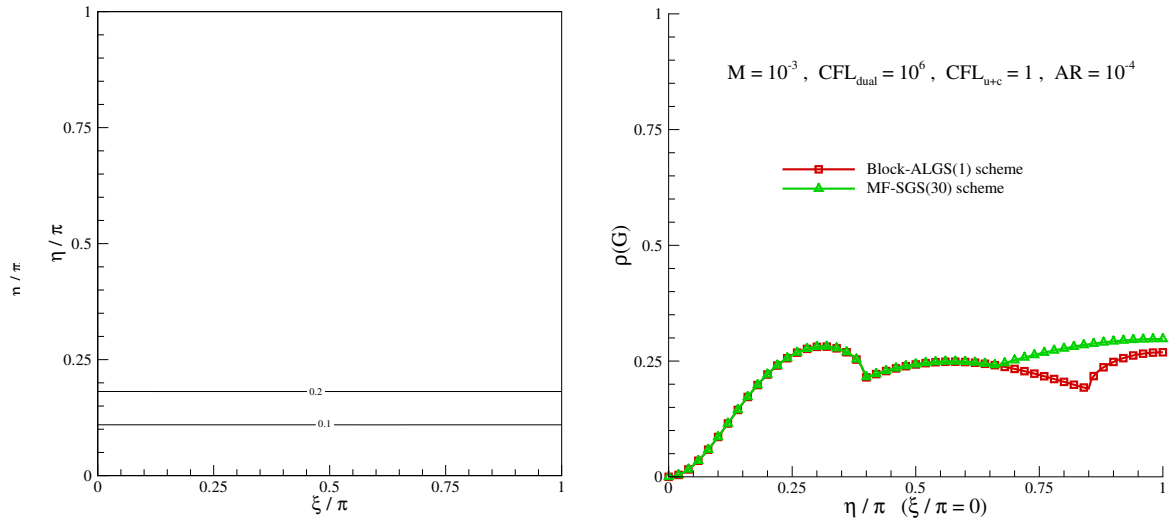


Figure 7.4.2: Left : Amplification factor of the MF-SGS(30) scheme at $M = 10^{-3}$, $CFL_{u+c} = 1$ and $CFL_\tau = 10^6$ with the unsteady preconditioning. Right : Profile of the amplification factor along the y-axis ($\xi = 0$) for the same conditions. Comparison between Block-ALGS(1) scheme and MF-SGS(30) scheme.

In fact, according to the chapter 3, we can easily explain this behaviour. Under such conditions, the physical time-derivative term is highly dominant for the x-modes and, consequently, the operator matrices H_1 and H_2 which define the SGS scheme (*cf.* expression 3.12.1) read :

$$H_1 \approx \frac{\Delta\tau}{\Delta t} Id \qquad \text{and} \qquad H_2 \approx \frac{\Delta\tau}{\Delta t} Id$$

Moreover, the implicit operator H and the explicit operator K read also :

$$H \approx \frac{\Delta\tau}{\Delta t} Id \qquad \text{and} \qquad K \approx \frac{\Delta\tau}{\Delta t} Id$$

As a result V_{SGS} is equal to zero and the amplification matrix of the SGS scheme (G_{SGS}) is the same than the Direct scheme's one (G_*). Besides, the amplification matrix of the Direct scheme is given by :

$$G_* = H^{-1} \cdot (H - K) \approx H^{-1} \cdot (\frac{\Delta\tau}{\Delta t} Id - \frac{\Delta\tau}{\Delta t} Id) \approx 0$$

Hence, the damping is maximum for the pure axial modes.

Let us now study the particle problem with high aspect ratio grids. A priori, we can expect that, as the physical time-derivative term is less dominant, the behaviour of the MF-SGS scheme becomes similar to the steady case. Figure 7.4.3 presents the results obtained for $AR = 10^{-1}$ and $AR = 10^{-3}$. We notice that, in the manner of the steady case, the amplification factor becomes stiff for pure longitudinal modes.

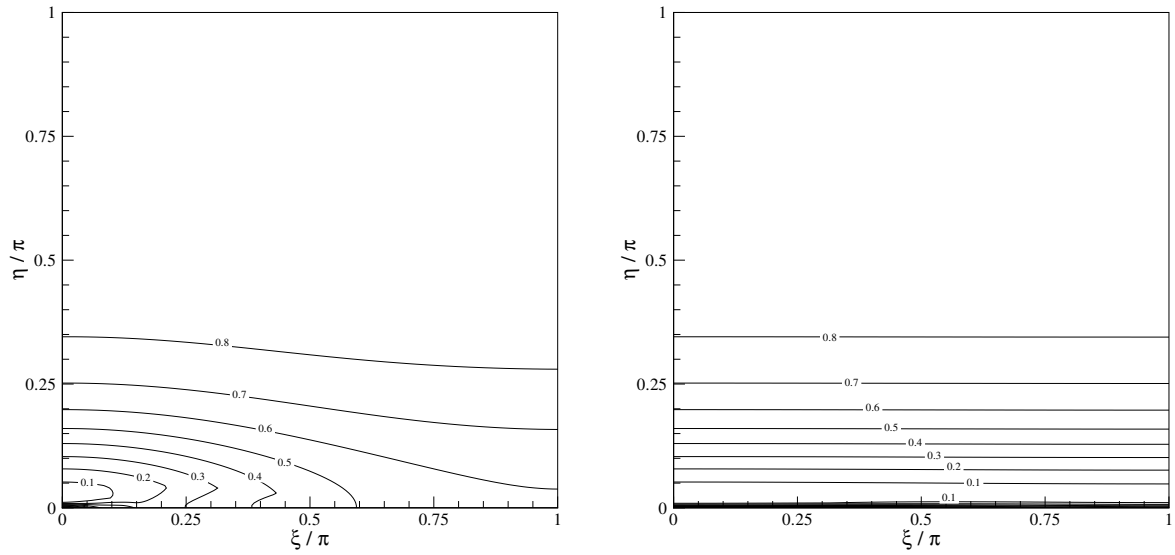


Figure 7.4.3: Amplification factor of the MF-SGS(30) scheme at $M = 10^{-3}$, $CFL_u = 1$ and $CFL_\tau = 10^6$ with the unsteady preconditioning. Left : $AR = 10^{-1}$. Right : $AR = 10^{-3}$.

However, the value of the aspect ratio implies that for longitudinal modes the physical time-derivative term becomes larger, and, as a result, the loss of efficiency is expected to be less important than for the steady case. In figure 7.4.4, we note that until $AR = 10^{-3}$, the MF-SGS method is competitive compared to the Block-ALGS scheme which is well-known to possess good properties in such conditions. Nevertheless, for $AR = 10^{-4}$, MF-SGS scheme becomes too stiff and, like for the steady case, it is unsuited for computing such problems. In spite of this weakness, we can conclude that the Matrix-Free method is a valuable choice to carry out unsteady computations with reasonable grid stretching.

7.5 Conclusion

In this chapter, we studied the performances of the Matrix-Free implicit method for the Euler unsteady equations solved using a dual-time stepping strategy. We described the difficulties related to the low-Mach number computation and we outlined a specific definition of the low-Mach preconditioner. Using this unsteady preconditioner, we showed that the Matrix-Free method coupled with SGS or PJ algorithm is highly efficient to damp the low-frequency modes for both acoustic and particle problems. Furthermore, we emphasized that the method is also versatile enough to deal with the intermediate cases that can arise when one uses high aspect ratio grids. The theoretical Von Neumann analysis enabled us to study the main features of the Matrix-Free implicit method such that we can conclude that this method represents a valuable choice to

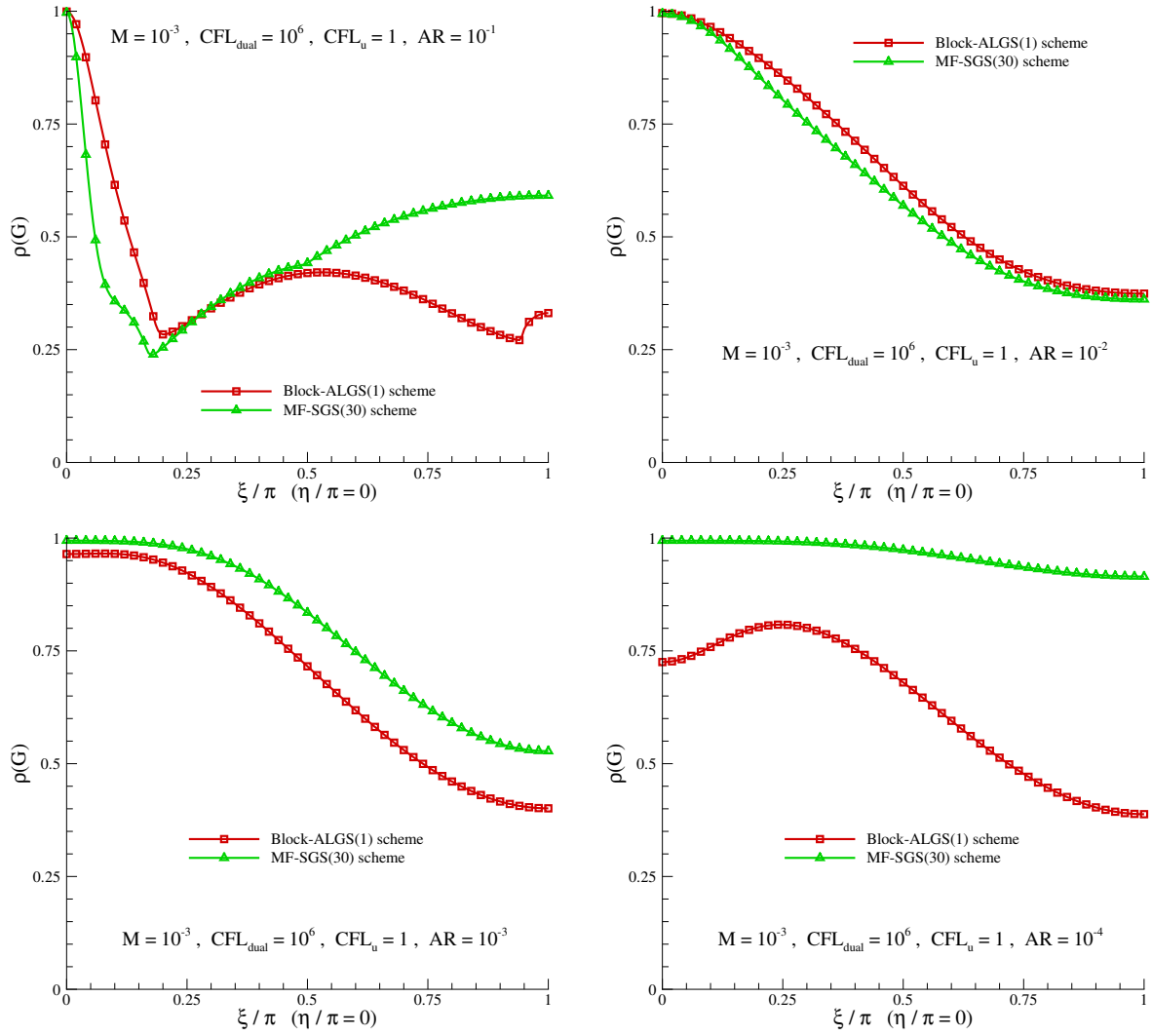


Figure 7.4.4: Amplification factor of the MF-SGS(30) scheme at $M = 10^{-3}$, $CFL_u = 1$ and $CFL_\tau = 10^6$ with the unsteady preconditioning. Right : $AR = 10^{-1}$. Left : $AR = 10^{-3}$.

compute unsteady flows. However, as mentioned earlier, we have to take care about the fact that the unsteady preconditioner can have a negative effect on the accuracy of the computation since, especially for acoustic problems, it approaches the no preconditioning formulation which implies a poor scale of the numerical dissipation. We will study all these properties in the numerical experiments.

Chapter 8

Basic Numerical Experiments

8.1 Introduction

This chapter deals with numerical experiments that have been carried out using a model code, with a two-folded objective :

- to confirm by practical computations the *a priori* analysis of intrinsic efficiency based on the study of the amplification factor for the steady and unsteady linearized Euler and Navier-Stokes equations,

- to complete the performance analysis of the proposed matrix-free treatment by taking into account the computational cost of the method as well as its memory requirements, thus gaining access to a preliminary assessment of the global efficiency provided by the matrix-free approach.

The use of a model code was retained for flexibility reason : this code solely developed for our own purpose proved to be a very useful numerical platform for carrying out a large number of numerical experiments with several variants of the implicit schemes under study before actually implementing in CEA's CAST3M general-purpose platform the best choice identified at the conclusion of this preliminary study. The model code is written using a finite-difference approach on Cartesian grids. It is applied to three test-cases, designed so as to allow the investigation of a wide range of configurations : namely an inviscid steady problem, both in the compressible and low-Mach regime (flow over a sine-bump), a viscous low-Mach steady problem (Poiseuille flow) and a viscous low-Mach unsteady problem (flow over an oscillating flat plate). For each case, we want to outline the different properties of the Matrix-Free method, mainly the convergence rate and the memory storage required, and situate these properties with respect to the ones provided by standard block-implicit treatments (representative of the available block-implicit treatment in CAST3M). The results that we are about to comment will enable us to justify (or not!) the valuableness of our method in order to implement it into the CAST3M code of the CEA.

8.2 Model platform framework

Let us present the governing equations and the non-dimensionalization that we applied in the model program. Unlike the process that we described in chapter 1, we choose the free-stream velocity as the reference speed. It follows that non-dimensional quantities are defined by :

$$t\vec{r} = \frac{\tilde{\vec{r}}}{\tilde{L}} \quad , \quad \rho = \frac{\tilde{\rho}}{\tilde{\rho}_\infty} \quad , \quad \vec{u} = \frac{\tilde{\vec{u}}}{\tilde{u}_\infty} \quad , \quad T = \frac{\tilde{T}}{\tilde{T}_\infty} \quad , \quad p = \frac{\tilde{p}}{\tilde{\rho}_\infty \tilde{u}_\infty^2} \quad ,$$
$$\mu = \frac{\tilde{\mu}}{\tilde{\mu}_\infty} \quad , \quad \lambda = \frac{\tilde{\lambda}}{\tilde{\lambda}_\infty} \quad , \quad t = \frac{t}{\tilde{L}/\tilde{u}_\infty} \quad , \quad E = \frac{\tilde{E}}{\tilde{u}_\infty^2} \quad , \quad H = \frac{\tilde{H}}{\tilde{u}_\infty^2} \quad ,$$

where subscript ∞ denotes the reference quantities. We can also define a reference sound speed in order to determine the reference Mach number :

$$c_\infty^2 = \gamma R T_\infty \quad \Rightarrow \quad M_\infty^2 = \frac{u_\infty^2}{\gamma R T_\infty}$$

Such choices lead to the following dimensionless equation of state :

$$p = \frac{1}{\gamma M_\infty^2} \rho T$$

The dimensionless Navier-Stokes equations that we implemented in our program are then the following :

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{u}) = 0 \\ \dfrac{\partial \rho \vec{u}}{\partial t} + \vec{\nabla} \cdot (\rho \vec{u}\ \vec{u}) + \vec{\nabla} p = \dfrac{1}{Re}\ (\vec{\nabla} \cdot \underline{\tau}) \\ \dfrac{\partial \rho E}{\partial t} + \vec{\nabla} \cdot (\rho \vec{u} H) = \dfrac{1}{Ec\ Pr\ Re}\ \vec{\nabla} \cdot (\mu \vec{\nabla} T) + \dfrac{1}{Re}\ \vec{\nabla} \cdot (\underline{\tau} \cdot \vec{u}) \end{cases}$$

where the primary dimensionless parameters are given by :

$$Re = \frac{\rho_\infty u_\infty L}{\mu_\infty} \quad , \qquad Pr = \frac{\mu_\infty C_p}{\lambda_\infty} \quad , \qquad Ec = \frac{u_\infty^2}{C_p T_\infty} = (\gamma - 1)\ M_\infty^2 \quad .$$

In order to prevent the round-off errors which can occur at very low-Mach number, we split, in the manner of [14], the pressure into two parts, namely a thermodynamic pressure of order $O(1/M^2)$ which remains constant with respect to the space dimension and a dynamic pressure of order $O(1)$ which varies in time and in space. Hence we have :

$$p(t,x) = P_0(t) + p'(t,x)$$

where p is the total pressure, P_0 is the thermodynamic pressure and p' is the dynamic pressure. Practical implementation in the "maquette" is inspired by the work of Sockol [73]. In the following test-cases, the thermodynamic pressure is constant and equal to the initial pressure :

$$P_0(t) = p_0 = \frac{1}{\gamma M_\infty^2}$$

It follows that :

$$p' = p - \frac{1}{\gamma M_\infty^2}$$

The density is obtained through the equation of state :

$$\rho = (1 + p'/p) \ / \ T$$

while the total energy reads :

$$\rho E = \frac{p_0 + p'}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2)$$

Hence, we can also define a reduced energy :

$$e' = \frac{p'}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2)$$

As a result the new set of conservative variables is the following :

$$w = (\rho,\ \rho u,\ \rho v,\ e')^T$$

In [40], we proved that such a correction enables the code to reach the machine zero even for Mach numbers close to 10^{-8}. Moreover, the quality of the solution can be also preserved.

Let us now describe some features that are common to all test-cases. First of all, we only deal with the Roe-Turkel scheme extended to higher accuracy thanks to the non-limited MUSCL extrapolation. Second, we evaluate the viscous fluxes thanks to a second-order centered formula. Explicit boundary conditions are used by default but, in some case, we will treat the boundary conditions implicitly.

8.3 Flow over a sine bump

8.3.1 Description

An inviscid flow in a channel is computed. The geometry of the channel consists of a rectangular domain $(x, y) \in [0; 4] \times [0; 1]$, with a straight upper wall and a curved lower wall, whose coordinates are given by :

$$\begin{cases} x < 1 & y = 0 \\ 1 \geq x \geq 3 & y = 0.1\ (1 - \cos[(x-1)\pi]) \\ x > 3 & y = 0 \end{cases}$$

In order to adapt this geometry to our Cartesian grid, the lower wall remains straight but, when $x \in [1; 3]$, the normal to the wall is computed as follows :

$$\begin{cases} n_x = -0.1 \cdot \pi \cdot \sin[(x-1)\pi] \\ n_y = 1 \end{cases}$$

Three grids are employed to carry out the studies, namely a coarse one (81×21), a medium one (121×31) and a fine one (161×41). For each grid, we solve the Euler equations for an ideal gas, with $\gamma = 1.4$. For $M_\infty \leq 0.5$ the flow remains subsonic, and hence isentropic and irrotational (because for higher Mach numbers one gets a transonic flow with a shock), and the Mach isolines are expected to be completely symmetrical with respect to the geometry (*i.e.* potential flow). Non-dimensional values are taken as follows : $\rho_0 = 1$, $u_0 = 1$, $v_0 = 0$, $P_0 = 1/\gamma M_\infty^2$. For non-low Mach number flows, that is when $M_\infty > 0.1$, we consider subsonic boundary conditions :

- at the inlet (left) we impose the external total enthalpy, the external entropy and the flow angle while we extrapolate the pressure from inside;

- at the outlet (right) we impose the static pressure while we extrapolate the density and the velocity from inside.

On the other hand, when the Mach number is small ($M_\infty \leq 0.1$), we modify the inlet conditions by considering "incompressible" inlet conditions. We impose the external density and velocity while we extrapolate the pressure from inside. In fact, we want to use the same boundary conditions for $M_\infty = 0.1$ and $M_\infty = 10^{-5}$ so that the comparisons between convergence rates are not influenced by the boundary effects. The choice of the incompressible conditions is truly justified for $M_\infty = 10^{-5}$ but is is of course debatable for $M_\infty = 0.1$. At any case, our objective in this chapter concerns more the performance properties of our scheme rather than the quality of the solution.

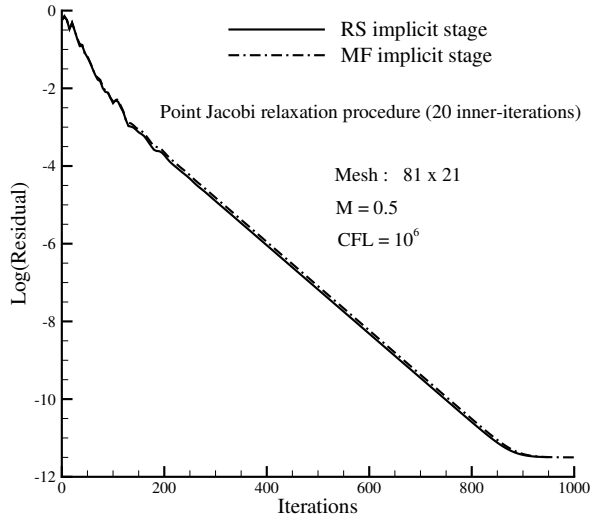
8.3.2 Objectives of the study

Here, we first want to study the impact of our Matrix-Free framework in terms of CPU time. Indeed, the Von Neumann analysis can tell us how the spectral radius simplification alters the convergence rate but it fails to inform us about the gain obtained by the simplified procedure that we described in chapter 4. For this reason, we created in the code an implicit stage named RS (RS standing for "Rayon Spectral" or Spectral Radius). This RS implicit stage is in fact a standard block implicit stage in which we replaced the numerical dissipation matrices by the identity matrix multiplied by their spectral radius. Of course, such an implicit stage has no practical interest since it combines the loss of efficiency due to the RS simplification plus the high cost per iteration of the block scheme. However, if one compares this implicit stage coupled with a Point Jacobi relaxation procedure with respect to the MF-PJ scheme, one can calculate the amount of CPU time saved by the Matrix-Free framework.

Besides, we want to emphasize the impact of the low-Mach preconditioning in terms of efficiency and accuracy. We especially insist on the independence of the convergence rate with respect to the Mach number when the inviscid preconditioning is employed. Such results are well-established now for the block schemes but they have to be confirmed for the Matrix-Free method. Our reference method will be the block-ALGS(1) scheme which is well-known to possess very good convergence properties. We will lead systematic comparisons between this scheme and our Matrix-Free procedure for a subsonic case at $M_\infty = 0.5$ and for a very low-Mach case at $M_\infty = 10^{-5}$. The MF scheme will be coupled with a SGS or a PJ relaxation procedure.

8.3.3 Analysis of the Matrix-Free framework

Let us consider a subsonic flow at $M_\infty = 0.5$. As mentioned in the previous paragraph, we deal with a RS implicit stage and the Matrix-Free implicit stage. Both methods are solved by the Point Jacobi relaxation procedure and they are employed to compute the subsonic flow on the coarse grid. The difference between these two stages stands in the fact that the Matrix-Free scheme minimizes the number of operations that are required to invert the system. The results are presented in figure 8.3.1. Both methods require the same number of iteration to reach the steady state, indicating that the Matrix-Free framework has no negative impact on the convergence rate. In [41], we showed the same results for different grid sizes and also for the Poiseuille flow. Besides, we notice in the summarizing table that the Matrix-Free framework enables us to save CPU time. Indeed, the time to converge is dramatically reduced with the MF scheme (from 72 seconds to 38 seconds). These first results emphasize that the Matrix-Free method that we developed possesses a very low cost per iteration and that the involved simplifications do not affect the convergence rate.



Method	Iterations	CPU
RS-PJ(20)	900	72 sec.
MF-PJ(20)	900	38 sec.

Figure 8.3.1: Impact of the Matrix-Free framework. Left : Convergence history of the residual on the coarse grids for RS-PJ(20) and MF-PJ(20) methods at $M_\infty = 0.5$ and $CFL = 10^6$. Right : Table summarizing the convergence properties of both methods, namely the number of iterations to reach the steady state and the associated CPU time.

8.3.4 Comparison Block scheme / MF scheme for subsonic flow

In this paragraph, we want to corroborate the results obtained with the Von Neumann analysis. Before leading comparisons, we have to define the cost of a numerical method. This cost can be expressed as follows :

$$C = N \cdot I_c$$

where C is the global cost, N is the number of iterations and I_c is the cost per iteration. In fact, the number of iterations can be related to the amplification factor of the method according to

the following expression :

$$(\rho(G))^N = \varepsilon$$

where $\rho(G)$ is the amplification factor and ε is the level of convergence reached after N iterations. Thus, one can predict the gap between two methods which reach the same level of convergence as follows :

$$\frac{\log(\rho(G_1))}{\log(\rho(G_2))} = \frac{N_2}{N_1}$$

This formula is true for one error mode in the Fourier space. In order to get an estimation of the gap for all the modes, we can make an average of the amplification factor :

$$\rho(G)_{ave} = \frac{1}{\pi^2} \int_0^\pi \int_0^\pi \rho(G(\xi, \eta))\ d\xi\ d\eta$$

In table 8.3.1, we present the average amplification factor at $M_\infty = 0.5$ of the methods that we want to compare, namely the BL-ALGS(1) scheme which is used as the reference one, the MF-SGS(10) scheme and the MF-PJ(30) scheme. In the same table, we also provide the theoretical ratio between all these methods. We notice that, in theory, the MF-SGS(10) scheme requires 2.5 times as many iterations as the BL-ALGS(1) scheme while the MF-PJ(30) scheme needs 2.8 times as many iterations. Such results emphasize that, if we want the MF-SGS(10) scheme to be competitive in terms of CPU time, its cost per iteration has to be at least 2.5 times less expensive than the BL-ALGS(1) scheme's one. This method is just a qualitative tool since the Von

Method	BL-ALGS(1)	MF-SGS(10)	MF-PJ(30)
$\rho(G)_{ave}$	0.54	0.78	0.80
N_i/N_{ref}	1	2.5	2.8

Table 8.3.1: Average amplification factor of several methods at $M_\infty = 0.5$: BL-ALGS(1) scheme, MF-SGS(10) scheme and MF-PJ(30) scheme. The theoretical ratio of iteration between all these methods is also provided, BL-ALGS(1) scheme is chosen as the reference.

Neumann analysis does not take into account the effects of the boundary conditions. Moreover, we know that the convergence rate is mainly driven by the performance at low wavenumbers, so we should increase the weight of the low-frequency in order to be more precise. Nevertheless, the estimation given by this analysis points out the effort that must be made so that the Matrix-Free method is competitive.
In figure 8.3.2, we present the diagonal profile of the amplification factors of these methods and the convergence history of the residuals on the fine grids. We notice that the hierarchy provided by the Von Neumann analysis remains the same for the numerical test-case : the BL-ALGS(1) scheme is the most efficient, it requires $N_1 = 720$ iterations to reach the zero machine; it takes $N_2 = 1320$ iterations for the MF-SGS(10) scheme to reach the same level and $N_3 = 1400$ iterations for the MF-PJ(30) scheme. We have then the following relative efficiency :

$$\frac{N_2}{N_1} = 1.83 \quad , \qquad \frac{N_3}{N_1} = 1.94 \quad .$$

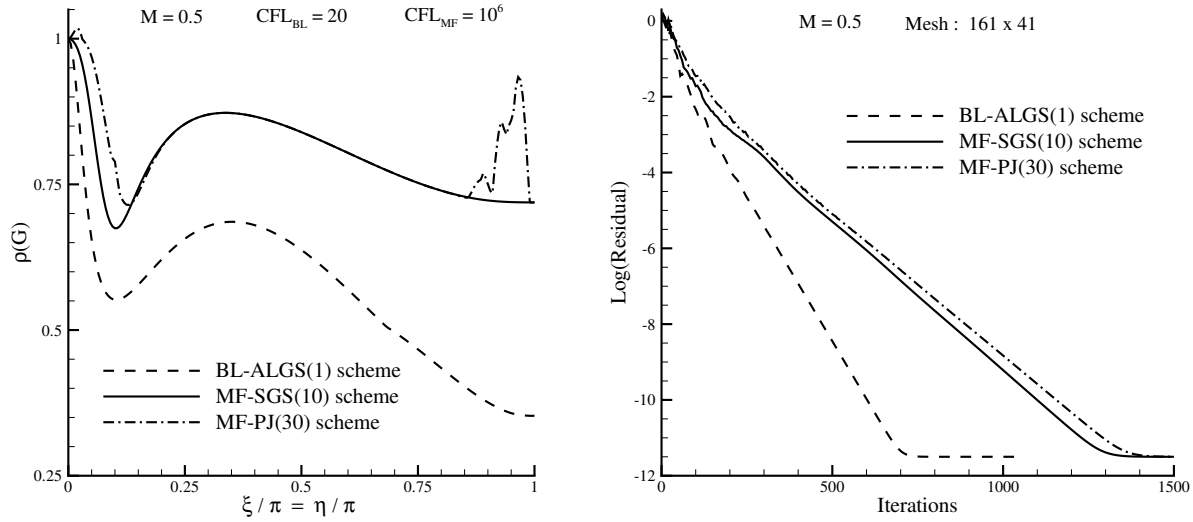


Figure 8.3.2: Comparison between BL-ALGS(1) scheme, MF-PJ(30) scheme and MF-SGS(10) scheme at $M_\infty = 0.5$. Left : Diagonal profile of the amplification factors. Right : Convergence history of the residuals on the fine grid.

Thus, the Von Neumann analysis overestimates the loss of intrinsic efficiency related to the Matrix-Free method. As explained earlier, the convergence rate relies mainly on the damping of the low wavenumbers. Indeed, the comparison of the amplification factors emphasizes that the gap between the different methods is less important over the low-frequency region. These results then give credence to such an assumption. Nevertheless, an important cost reduction is still necessary in order to make the Matrix-Free method competitive. The framework that we present in chapter 4 ensures such a reduction. The results obtained at $M_\infty = 0.5$ are presented in Table 8.3.2. The last column is related to the cost per point per iteration, we denote it CPPPI. The CPPPI remains quite constant with respect to the grid size, it is therefore very useful to drive comparisons. For the fine grid, the relative costs of the methods that we want to compare are the following :

$$\frac{CPPPI_1}{CPPPI_2} = 2.52 \quad , \qquad \frac{CPPPI_1}{CPPPI_3} = 2.07 \quad .$$

The Matrix-Free framework enables to reduce dramatically the CPPPI. Hence, the Matrix-Free method coupled with the SGS algorithm or the PJ procedure becomes globally more competitive than the Block scheme. Let us now focus on each technique. As the grid becomes finer, the MF-PJ scheme requires a larger amount of inner-iterations in order to remain efficient. This is in good agreement with the Von Neumann analysis, the MF-PJ algorithm needs many inner-iterations so as to damp correctly the low-frequency error modes. However, we observe some instabilities for very low wavenumbers while the scheme is stable in practice. The grid is perhaps not fine enough to capture these low-frequency modes, at any case this is another reason to increase the number of sub-iterations. On the other hand, the MF-SGS scheme remains very efficient for all grids with only 10 inner-iterations. This is also confirmed by the Von Neumann results. In the Table 8.3.2, we also provide the results obtained with the RS implicit stage for the coarse grid. It is interesting to note that when it is coupled with the ALGS(1) relaxation procedure, this method is the most efficient among the MF-type techniques.

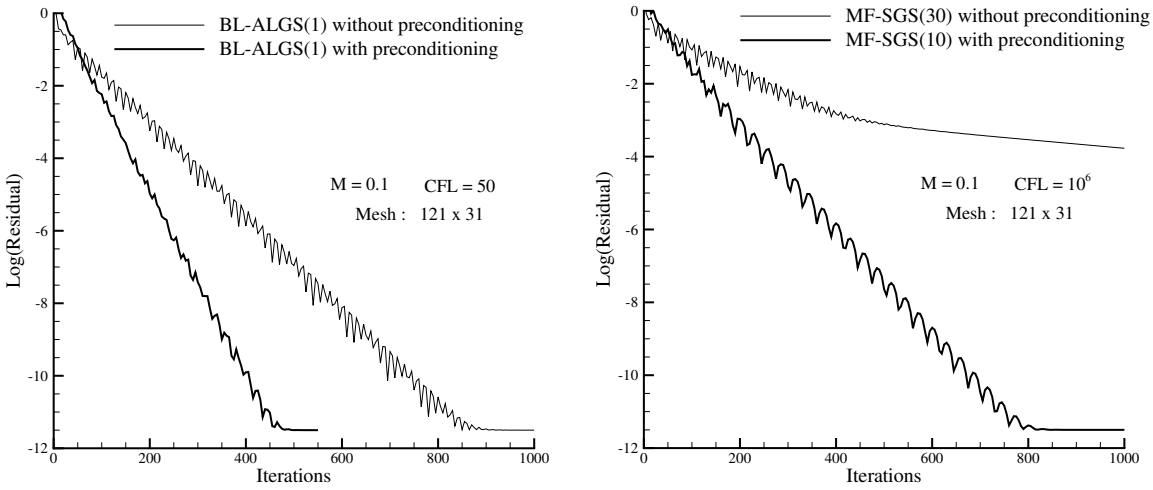
Hence, the loss of intrinsic efficiency inherent in the spectral radius simplification is balanced by the Matrix-Free implicit framework that we outlined in chapter 4. Indeed, the cost per iteration is strongly reduced so that the global cost becomes highly competitive compare to the classical block scheme. Moreover, we can expect a dramatically decrease of the memory requirements, thus, the Matrix-Free method is truly interesting to compute subsonic inviscid flows. Let us now carry out studies over the low-Mach regime.

NBEL	Method	CFL	Iterations	CPU	CPI	CPPPI
1701	BL-ALGS(1)	20	**400**	53 s.	0.13	$7.8 \cdot 10^{-5}$
	RS-ALGS(1)	10^6	710	83 s.	0.12	$6.9 \cdot 10^{-5}$
	RS-PJ(20)	10^6	900	72 s.	0.08	$4.7 \cdot 10^{-5}$
	MF-PJ(10)	10^6	1950	53 s.	0.03	$\mathbf{1.6} \cdot 10^{-5}$
	MF-PJ(20)	10^6	900	**38 s.**	0.04	$2.5 \cdot 10^{-5}$
	MF-PJ(30)	10^6	830	48 s.	0.06	$3.4 \cdot 10^{-5}$
	MF-SGS(10)	10^6	800	40 s.	0.05	$3 \cdot 10^{-5}$
	MF-SGS(20)	10^6	750	66 s.	0.09	$5.1 \cdot 10^{-5}$
	MF-SGS(30)	10^6	730	92 s.	0.17	$7.4 \cdot 10^{-5}$
3751	BL-ALGS(1)	20	**550**	169 s.	0.31	$8.2 \cdot 10^{-5}$
	MF-PJ(10)	10^6	NC	-	-	-
	MF-PJ(20)	10^6	1250	133 s.	0.11	$\mathbf{2.8} \cdot 10^{-5}$
	MF-PJ(30)	10^6	1120	165 s.	0.15	$3.9 \cdot 10^{-5}$
	MF-SGS(10)	10^6	1070	**130 s.**	0.12	$3.2 \cdot 10^{-5}$
	MF-SGS(20)	10^6	1000	215 s.	0.21	$5.7 \cdot 10^{-5}$
	MF-SGS(30)	10^6	970	300 s.	0.31	$8.2 \cdot 10^{-5}$
6601	BL-ALGS(1)	20	**720**	396 s.	0.55	$8.3 \cdot 10^{-5}$
	MF-PJ(20)	10^6	3000	581 s.	0.19	$\mathbf{2.9} \cdot 10^{-5}$
	MF-PJ(30)	10^6	1400	370 s.	0.26	$4 \cdot 10^{-5}$
	MF-SGS(10)	10^6	1320	**290 s.**	0.22	$3.3 \cdot 10^{-5}$
	MF-SGS(20)	10^6	1250	482 s.	0.39 s.	$5.8 \cdot 10^{-5}$
	MF-SGS(30)	10^6	1200	667 s.	0.568	$8.4 \cdot 10^{-5}$

Table 8.3.2: Inviscid steady flow over a sinebump at $M_\infty = 0.5$. Regular Cartesian grids with $AR = 1$. The computations were done with a Pentium 4, 2.4 GHz, 1GB RAM memory. NBEL is the number of points. CPU is the time required to converge. CPI is the cost per iteration while CPPPI is the cost per point per iteration.

8.3.5 Comparisons in the low-Mach regime

Before performing comparisons, we want to outline the necessity of the preconditioning in order to preserve efficiency and accuracy. Furthermore, we will emphasize the independence of the convergence rate with respect to the Mach number when the scheme is preconditioned.



PREC	Method	CFL	Iterations	CPU	CPI	CPPPI
No	BL-ALGS(1)	50	850	222 s.	0.26	$6 \cdot 10^{-5}$
	MF-PJ(30)	10^6	NC	-	-	-
	MF-SGS(10)	10^6	NC	-	-	-
Yes	BL-ALGS(1)	50	**470**	144 s.	0.31	$8.2 \cdot 10^{-5}$
	MF-PJ(30)	10^6	990	144 s.	0.15	$3.9 \cdot 10^{-5}$
	MF-SGS(10)	10^6	790	**96 s.**	0.12	**3.2** $\cdot 10^{-5}$

Figure 8.3.3: Inviscid steady flow over a sinebump at $M_\infty = 0.1$ for the medium grid (3751 points). Regular Cartesian grid with $AR = 1$. Top-Left : Convergence history of the residual for the BL-ALGS scheme with and without preconditioning. Top-Right : Convergence history of the residual for the MF-SGS scheme with and without preconditioning. Bottom : Table summarizing the performance of the preconditioned and unpreconditioned schemes at $M_\infty = 0.1$.

In figure 8.3.3, we present the results obtained with the preconditioned and the non-preconditioned schemes at $M_\infty = 0.1$ on the medium grid. As expected, we clearly note that the preconditioned schemes are faster than the unpreconditioned ones. As forecasted by the Von Neumann analysis, we remark that the Matrix-Free schemes without preconditioning possess poor convergence rate. We observe also that for the same grid, MF preconditioned schemes are

better at $M_\infty = 0.1$ than at $M_\infty = 0.5$ since the number of iterations required to reach the machine zero is smaller. This behaviour confirms the results of the chapter 5, the spectral radius simplification is less penalizing at low-Mach number since the eigenvalues of the preconditioned system are very close.

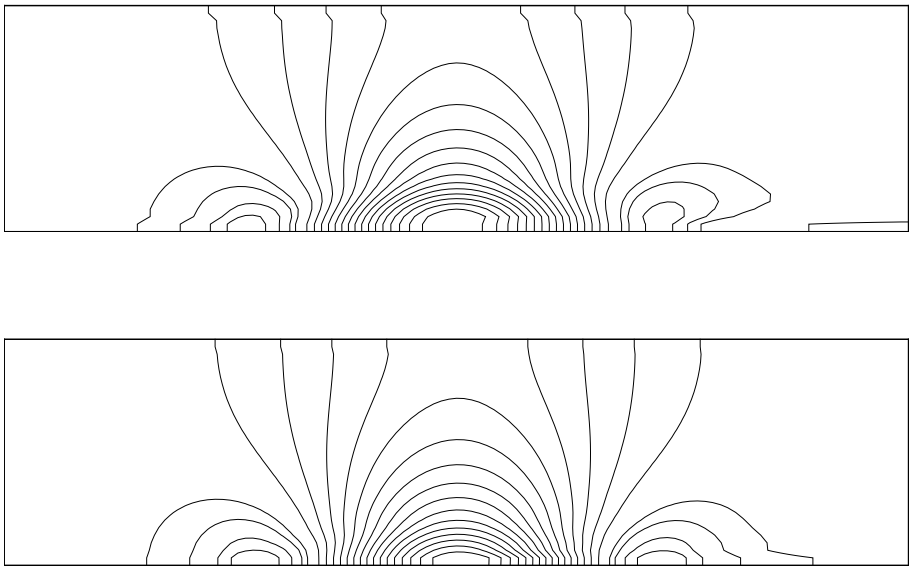


Figure 8.3.4: Isolines of the Mach number ($M_\infty = 0.1$, mesh (121×31)). Top : Unpreconditioned scheme. Bottom : Preconditioned scheme.

In figure 8.3.4, we draw the isolines of the Mach number at $M_\infty = 0.1$ on the medium grid. We note that, even if they converge to machine zero, the unpreconditioned schemes provide an incorrect solution. This well-known result is due to the over-dissipation of the numerical scheme which arises when the system is ill-conditioned (at low-Mach number for instance).

Another aspect of the low-Mach preconditioning's effects is the independence of the convergence rate with respect to the Mach number. In order to emphasize this property, we make a comparison between all the methods on the medium grid at $M_\infty = 0.1$ and $M_\infty = 10^{-5}$. The results are reported in table 8.3.3. As expected, the convergence rates of the different methods do not vary much for both Mach numbers. We can therefore carry out all the comparisons for a unique Mach number. We choose a quasi-incompressible case, namely $M_\infty = 10^{-5}$.
In the manner of the previous section, we present in figure 8.3.5 the diagonal profile of the amplification factors of different methods, namely the BL-ALGS(1) scheme, the MF-SGS(10) scheme and the MF-PJ(30) scheme. We also draw the convergence history of their residuals. Here again, the hierarchy provided by the Von Neumann analysis remains the same for the numerical experiment : the Block scheme is the most efficient in terms of iteration while the MF-SGS scheme is the fastest to reach the machine zero in terms of CPU time. These results are confirmed on all grids (*cf.* Table 8.3.4). As for the subsonic case, the loss of intrinsic

MACH	Method	CFL	Iterations	CPU	CPI	CPPPI
10^{-1}	BL-ALGS(1)	50	**470**	144 s.	0.31	$8.2 \cdot 10^{-5}$
	MF-PJ(30)	10^6	990	144 s.	0.15	$3.9 \cdot 10^{-5}$
	MF-SGS(10)	10^6	790	**96 s.**	0.12	**3.2** $\cdot 10^{-5}$
10^{-5}	BL-ALGS(1)	50	**470**	144 s.	0.31	$8.2 \cdot 10^{-5}$
	MF-PJ(30)	10^6	970	141 s.	0.15	$3.9 \cdot 10^{-5}$
	MF-SGS(10)	10^6	780	**95 s.**	0.12	**3.2** $\cdot 10^{-5}$

Table 8.3.3: Inviscid steady flow over a sinebump at $M_\infty = 0.1$ and at $M_\infty = 10^{-5}$ for the medium grid (3751 points). Regular Cartesian grid with $AR = 1$. NBEL is the number of points. CPU is the time required to converge. CPI is the cost per iteration while CPPPI is the cost per point per iteration.

efficiency inherent in the spectral radius simplification is balanced by the Matrix-Free implicit framework. The reduction of the cost per iteration enables the MF scheme to recover a very good global efficiency. Consequently, the Matrix-Free method is a valuable alternative to compute inviscid low-Mach number flow. Besides, the Matrix-Free implicit method has another advantage compare to the Block schemes, namely the reduction of the memory storage.

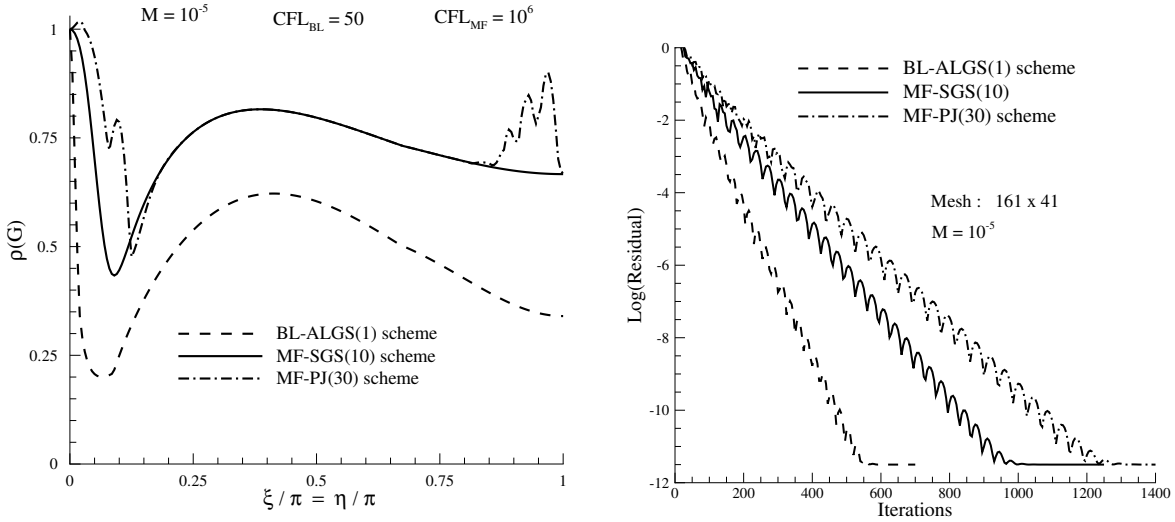


Figure 8.3.5: Comparison between BL-ALGS(1) scheme, MF-SGS(10) scheme and MF-PJ(30) scheme at $M_\infty = 10^{-5}$. Left : Diagonal profile of the amplification factors. Right : Convergence history of the residuals on the fine grid.

NBEL	Method	CFL	Iterations	CPU	CPI	CPPPI
1701	BL-ALGS(1)	50	**370**	49 s.	0.13	$7.8 \cdot 10^{-5}$
	MF-PJ(30)	10^6	700	40 s.	0.06	$3.4 \cdot 10^{-5}$
	MF-SGS(10)	10^6	550	**27 s.**	0.05	**3** $\cdot 10^{-5}$
3751	BL-ALGS(1)	50	**470**	144 s.	0.31	$8.2 \cdot 10^{-5}$
	MF-PJ(30)	10^6	970	141 s.	0.15	$3.9 \cdot 10^{-5}$
	MF-SGS(10)	10^6	780	**95 s.**	0.12	**3.2** $\cdot 10^{-5}$
6601	BL-ALGS(1)	50	**560**	308 s.	0.55	$8.3 \cdot 10^{-5}$
	MF-PJ(30)	10^6	1230	325 s.	0.26	$4 \cdot 10^{-5}$
	MF-SGS(10)	10^6	960	**208 s.**	0.22	**3.3** $\cdot 10^{-5}$

Table 8.3.4: Inviscid steady flow over a sinebump at $M_\infty = 10^{-5}$. Regular Cartesian grids with $AR = 1$. The computations were done with a Pentium 4, 2.4 GHz, 1GB RAM memory. NBEL is the number of points. CPU is the time required to converge. CPI is the cost per iteration while CPPPI is the cost per point per iteration.

8.3.6 Memory storage

The Matrix-Free implicit framework that we developed in Chapter 4 was asked by the French Atomic Energy Commission (CEA) in order to reduce dramatically the requirements in memory storage of the implicit schemes. The objective of such a reduction is to carry out numerical simulation of complex industrial problems involving grids with a very large amount of points. In this paragraph, we compare, for all grids, the memory requirements of each method with respect to the explicit scheme. The results are presented in Table 8.3.5. We note that the MF-SGS scheme is a bit more gluttonous with respect to the MF-PJ scheme. It is due to the fact that we need to store the states which are evaluated during each sweep so as to compute the next ones. On the other hand, the MF-PJ procedure does not need to store anything, this is the reason why it is the cheapest implicit method in terms of memory storage. As expected, the Block scheme's requirements are the greatest. It needs 110 % more ressources than the explicit scheme while the MF schemes require just 60 % more. Hence, the Matrix-Free framework enables to reduce of about half the memory requirements of the implicit scheme. Besides, as shown in the previous sections, the Matrix-Free implicit method is also very competitive in terms of CPU time to simulate low-Mach inviscid flows.

	Coarse Grid 81×21		Medium Grid 121×31		Fine Grid 161×41	
	Memory storage (MB)	relative size	Memory storage (MB)	relative size	Memory storage (MB)	relative size
Explicit scheme	3.1	1	6.6	1	12.2	1
MF-PJ scheme	4.7	1.52	10.1	1.53	19	1.56
MF-SGS scheme	4.8	1.55	10.5	1.59	19.6	1.61
BL-ALGS scheme	6.7	2.16	14.7	2.23	25.8	2.11

Table 8.3.5: Memory storage for all the methods with respect to the grid size. The explicit scheme is taken as the reference so as to evaluate the relative sizes.

8.3.7 Conclusions

The Matrix-Free implicit method is then a valuable alternative to simulate the inviscid steady flows. Indeed, even if its intrinsic efficiency is poor, its cost per iteration is so cheap that the method is globally more efficient than a Block implicit scheme coupled with a ALGS relaxation procedure. Furthermore, the Matrix-Free framework enables to reduce dramatically the need of memory storage. Such results have to be confirmed for viscous flows. That is the subject of the next section.

8.4 Poiseuille flow

8.4.1 Description

A viscous flow in straight channel is computed. The geometry of the channel consists of a rectangular domain $(x, y) \in [0; 10] \times [-1; 1]$.

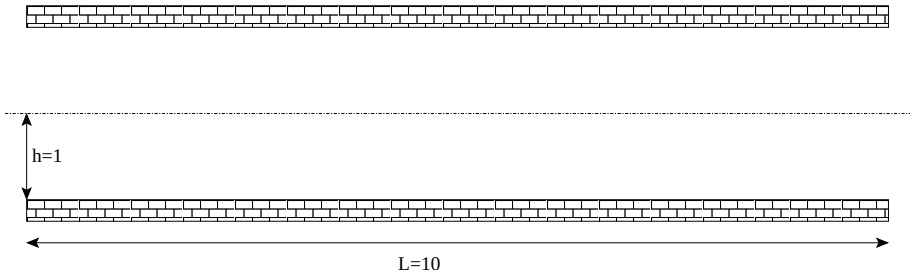


Figure 8.4.1: Computational domain for the Poiseuille flow.

For an incompressible steady flow without temperature and gravitational effects, the pressure function is linear and the Navier-Stokes equations reduce to the momentum one (*cf.* [66] for more details) :

$$\frac{1}{Re} \cdot \frac{d^2u}{dy^2} = \frac{dp}{dx}$$

where the Reynolds number is given by :

$$Re = \frac{\rho_\infty \, u_\infty \, l_\infty}{\mu_\infty}$$

with $l_\infty = h$ which is chosen as the length scale of the problem. Given to the no-slip conditions on each wall, the exact solution is then :

$$u(y) = -\frac{Re}{2} \cdot \frac{dp}{dx} \cdot (1 - y^2)$$

The velocity profile is therefore parabolic and the maximum value is obtained on the axis of the channel :

$$u_{\max} = -\frac{Re}{2} \cdot \frac{dp}{dx}$$

The average velocity is easily computed through the following expression :

$$u_{ave} \;=\; \frac{1}{2} \int_{-1}^{1} u_{\max} \cdot (1 - y^2) \; dy \;=\; \frac{2}{3} \cdot u_{\max}$$

Setting arbitrarily the average velocity equal to unity, the incompressible exact solution is then given by :

$$u(y) = 1.5 \cdot (1 - y^2)$$

We expect to obtain the same parabolic profile at the outlet of the channel. Non-dimensional values are taken as follows : $\rho_0 = 1$, $u_0 = 1$, $v_0 = 0$, $P_0 = 1/\gamma M_\infty^2$, $M_\infty = 10^{-5}$, $Re = 15$. We consider the incompressible boundary conditions :

- at the inlet (left) we impose the density, the velocity and its parabolic profile, and we extrapolate the pressure from inside;
- at the outlet (right) we impose the static pressure while we extrapolate the density and the velocity from inside.

The upper and lower walls are considered adiabatic and the no-slip condition is applied.

8.4.2 Objectives of the study

In chapter 6, we studied the impact of the viscous spectral radius simplification with the Von Neumann analysis. We outlined that for low-Mach and low-Reynolds flow, the Matrix-Free scheme was quite efficient providing that the aspect ratio remains of order one. We also presented an enhancement of the viscous preconditioning which enables the Matrix-Free method to remain competitive especially for the low-frequency modes. For this test-case, we want to confirm the results of the Von Neumann analysis. First, we will study for $AR = 1$ the performances of the implicit methods that we have already compared for the sinebump, namely the BL-ALGS scheme, the MF-SGS scheme and the MF-PJ scheme. Then, we will emphasize the loss of efficiency of the MF schemes when the aspect ratio increases. Finally, we will provide some practical solutions in order to recover efficiency.

8.4.3 Comparison Block scheme / MF scheme for $AR = 1$

In this paragraph, we consider a low-Mach viscous flow ($M_\infty = 10^{-5}$ and $Re = 15$) and we compute the solution on three grids with an aspect ratio equal to unity :

- 101×21 (coarse);
- 201×41 (medium);
- 301×61 (fine).

These three sizes of grid correspond to three different cell Reynolds numbers. Using the definition (6.2.1), we compute $Re_m^c = 0.76$, $Re_m^m = 0.38$ and $Re_m^f = 0.25$. For the fine grid, *ie.* for the case for which the viscous effects are the most important, the acoustic Reynolds number is given by :

$$Re_c^f = \frac{Re_m^f}{M_\infty} = 2.5 \cdot 10^4 >> 1 \ .$$

We can therefore expect that the spectral radius simplification would be not too penalizing. Let us now consider the fine grid. As every implicit scheme is coupled with the same explicit stage, all the numerical solutions are the same. In figure 8.4.2, we draw the distribution of the velocity at the outlet of the channel and we compare it with the exact incompressible solution.
The amplification factor at the reference state and the convergence history of the residuals of the three implicit schemes are presented in figure 8.4.3. We note that for the very low wavenumbers, the Matrix-Free schemes provide an as good damping as the BL-ALGS(1) scheme, indicating a good convergence rate. The convergence history of the residuals confirms that Matrix-Free schemes (MF-SGS(10) and MF-PJ(40)) are as efficient as the BL-ALGS(1). It is noticeable that both MF schemes provide the same efficiency. However, as the MF-PJ scheme requires 40

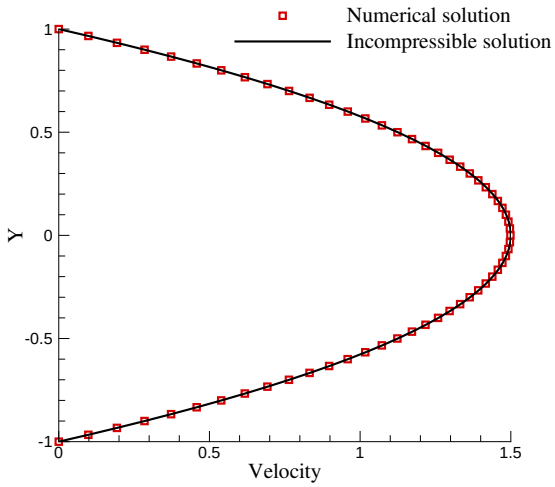


Figure 8.4.2: Distribution of the velocity at the outlet of the channel at $M_\infty = 10^{-5}$, $Re = 15$ and for the fine grid (301×61). Square symbols represent the numerical solution while the solid line stands for the exact incompressible solution.

inner-iterations, its cost per iteration is higher than the MF-SGS scheme with only 10 inner-sweeps. The results in Table 8.4.1 emphasize the good intrinsic efficiency of the MF schemes for all the grids. This is in good agreement with the observations of Chapter 6, that is the MF implicit method is highly competitive to compute viscous flows over grids with aspect ratio close to unity. Indeed, the MF method provides high intrinsic efficiency and a low cost per iteration, it is therefore very interesting to compute such flows. Nevertheless, computing viscous flows often requires stretched grids and we saw in Chapter 6 that the Matrix-Free method coupled with SGS or PJ relaxation procedure becomes less efficient for such cases. We will study this issue in the next paragraph.

8.4.4 Comparison for stretched grids

In order to carry out comparisons on stretched grids, we used three different grids. The two first ones are built using constant space steps while the last one was made using variable space steps in the y-direction :

- 21×41 with $AR = \delta x/\delta y = 10$;
- 21×81 with $AR = \delta x/\delta y = 20$;
- 21×41 with $AR_{\max} = \delta x/\delta y_{\min} = 50$ and $AR_{\min} = \delta x/\delta y_{\max} = 2$.

Convergence histories of the residual obtained on each grid are presented in figure 8.4.4. As expected, the efficiency of the MF-SGS scheme decrease as the aspect ratio becomes bigger. This loss of intrinsic efficiency cannot be balanced by the low cost per iteration of the method as it is shown in Table 8.4.2. Hence, the Matrix-Free implicit scheme is not a good alternative in such cases. On the other hand, the BL-ALGS scheme remains efficient for all aspect ratio grids,

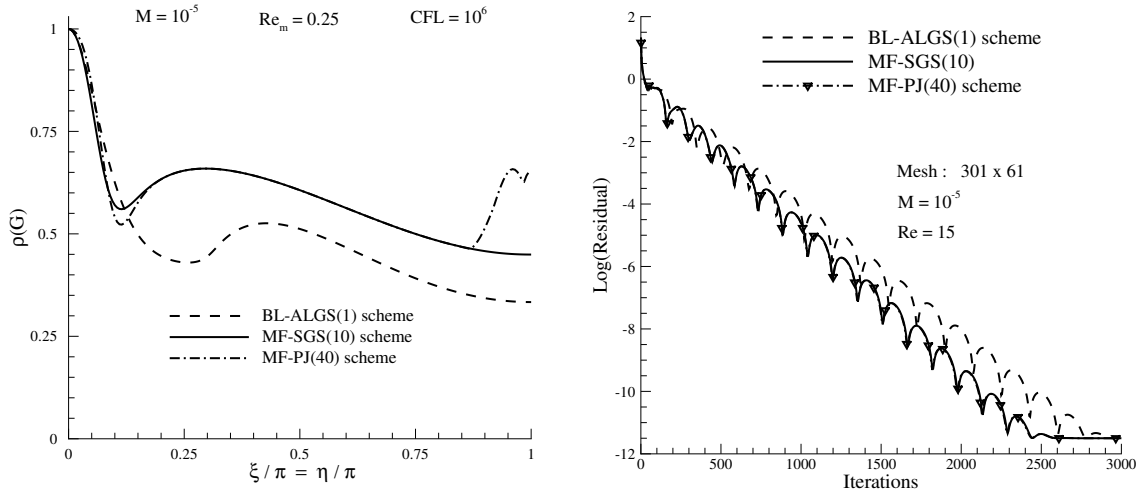


Figure 8.4.3: Comparison between BL-ALGS(1) scheme, MF-SGS(10) scheme and MF-PJ(40) scheme at $M_\infty = 10^{-5}$ and $Re = 15$. Left : Diagonal profile of the amplification factors. Right : Convergence history of the residuals on the fine grid.

NBEL	Method	CFL	Iterations	CPU	CPI	CPPPI
2121	BL-ALGS(1)	10^6	**450**	75 s.	0.17	$7.9 \cdot 10^{-5}$
	MF-PJ(30)	10^6	690	53 s.	0.077	$3.6 \cdot 10^{-5}$
	MF-SGS(10)	10^6	570	**37 s.**	0.065	**3** $\cdot 10^{-5}$
8241	BL-ALGS(1)	10^6	**1300**	903 s.	0.7	$8.4 \cdot 10^{-5}$
	MF-PJ(30)	10^6	1700	578 s.	0.34	$4.1 \cdot 10^{-5}$
	MF-SGS(10)	10^6	1310	**360 s.**	0.27	**3.3** $\cdot 10^{-5}$
18361	BL-ALGS(1)	10^6	2780	4375 s.	1.57	$8.6 \cdot 10^{-5}$
	MF-PJ(40)	10^6	**2450**	2383 s.	0.97	$5.3 \cdot 10^{-5}$
	MF-SGS(10)	10^6	**2450**	**1538 s.**	0.63	**3.4** $\cdot 10^{-5}$

Table 8.4.1: Viscous steady flow in a straight channel at $M_\infty = 10^{-5}$ and $Re = 15$. Regular Cartesian grids with $AR = 1$. NBEL is the number of points. CPU is the time required to converge. CPI is the cost per iteration while CPPPI is the cost per point per iteration.

this is in good agreement with the results of Venkateswaran *et al.* [12]. In order to circumvent this issue, we add the implicit treatment of the boundary conditions (IBC) that we described in

chapter 4. Unfortunately, it does not enable the method to converge faster when constant space steps are used. However, when the aspect ratio is chosen such as to be big close to the wall and small far from it, the MF-SGS scheme with IBC recovers a quite good intrinsic efficiency. Moreover, such a treatment does not increase the cost per iteration of the method, hence, the Matrix-Free scheme can remain very competitive. In figure 8.4.4, we also present the distribution of the velocity at the outlet of the channel obtained over each stretched grid. We note that the accuracy is quite the same for every mesh and that it is relatively poor. This lack of accuracy is only due to the small amount of points along the x-direction but it was not the main goal of the study in this section.

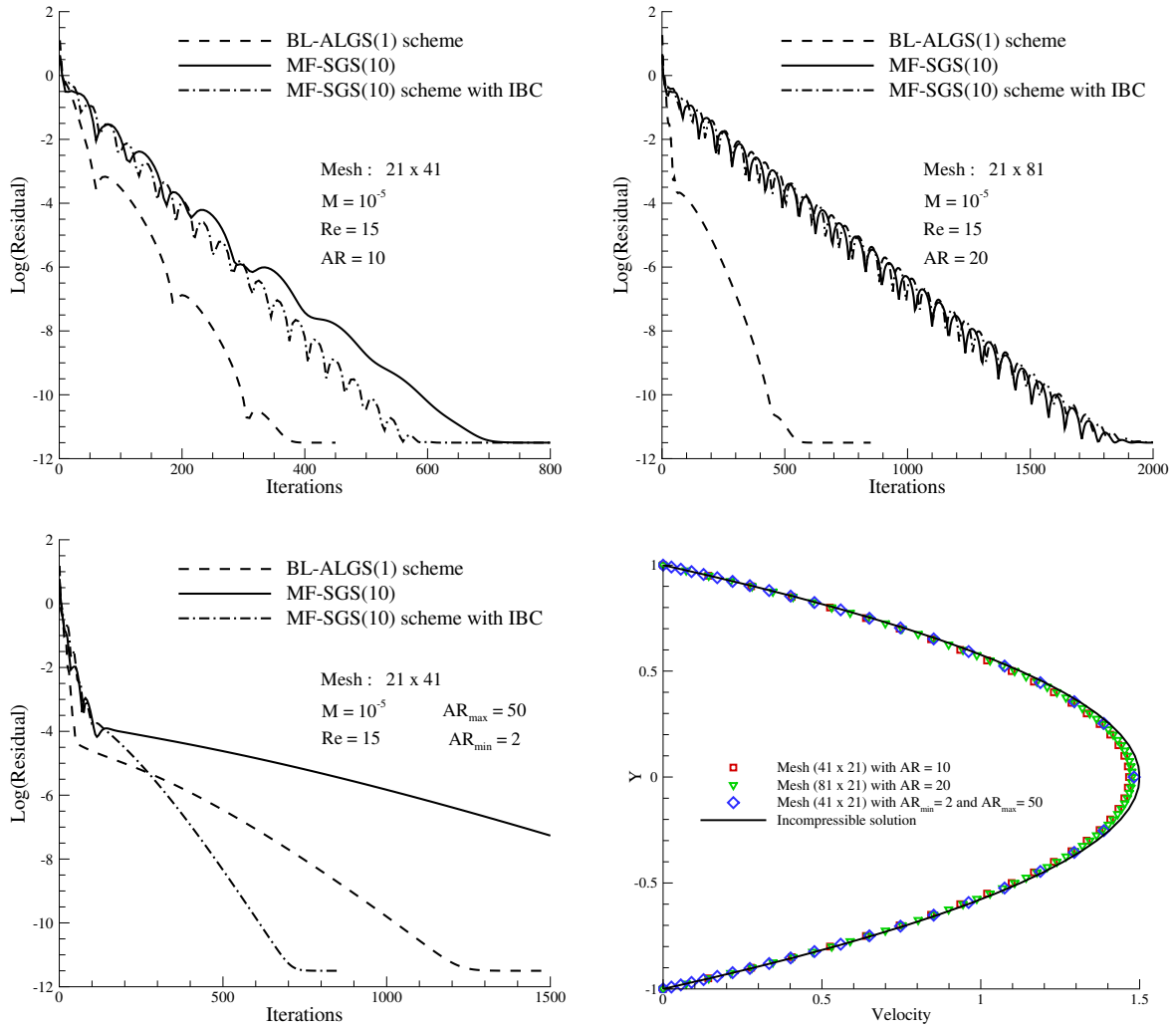


Figure 8.4.4: Convergence histories of the residuals for the BL-ALGS(1) scheme, the MF-SGS(10) scheme and the MF-SGS(10) scheme with an implicit treatment of the boundary conditions (IBC). $M_\infty = 10^{-5}$ and $Re = 15$. Top-Left : $AR = 10$. Top-Right : $AR = 20$. Bottom-Left : $AR_{\max} = 50$ and $AR_{\min} = 2$. Bottom-Right : Distribution of the velocity at the outlet of the channel obtained over each stretched grid.

Mesh	Method	CFL	Iterations	CPU	CPI	CPPPI
21×41	BL-ALGS(1)	10^6	**380**	25 s.	0.067	$7.7 \cdot 10^{-5}$
$AR = 10$	MF-SGS(10)$_{exp}$	10^6	720	17 s.	0.024	**2.7** $\cdot 10^{-5}$
	MF-SGS(10)$_{imp}$	10^6	590	**14 s.**	0.024	**2.7** $\cdot 10^{-5}$
21×81	BL-ALGS(1)	10^6	**570**	**78 s.**	0.14	$8.1 \cdot 10^{-5}$
$AR = 20$	MF-SGS(10)$_{exp}$	10^6	1850	101 s.	0.055	**3.2** $\cdot 10^{-5}$
	MF-SGS(10)$_{imp}$	10^6	1900	104 s.	0.055	**3.2** $\cdot 10^{-5}$
21×41	BL-ALGS(1)	10^6	1240	84 s.	0.068	$7.9 \cdot 10^{-5}$
$AR_{\max} = 50$	MF-SGS(10)$_{exp}$	10^6	2820	67 s.	0.024	**2.8** $\cdot 10^{-5}$
$AR_{\min} = 2$	MF-SGS(10)$_{imp}$	10^6	**720**	**17 s.**	0.024	**2.8** $\cdot 10^{-5}$

Table 8.4.2: Viscous steady flow in a straight channel at $M_\infty = 10^{-5}$ and $Re = 15$. Regular stretched grids with $AR = 10$, $AR = 20$. Irregular stretched grid with $AR_{\max} = 50$ and $AR_{\min} = 2$.

8.4.5 Memory storage

The Matrix-Free framework is limited to compute flows with very stretched grids. However, in terms of memory storage, the method is still a valuable choice since it requires only 60 % of additional memory with respect to the explicit scheme while the Block scheme needs about 110 % more. There is then a strong reduction of the memory requirement which can justify the use of such a method to compute viscous flows.

8.4.6 Conclusions

As forecasted by the Von Neumann analysis in chapter 6, the Matrix-Free schemes provide very good intrinsic efficiency to solve low-Mach viscous flows with grids whose aspect ratios are close to unity. On the other hand, because of the use of the SGS or PJ relaxation procedure, the method becomes stiff as the aspect ratio increases and the low cost per iteration inherent in the Matrix-Free framework is not sufficient to balance the loss of efficiency. This problem can be partially circumvented if one employs both an implicit treatment of the boundary conditions and a non-constant aspect ratio (typically big close the wall and smaller far from it). For such conditions, the Matrix-Free scheme recovers a very good intrinsic efficiency. Besides, the Matrix-Free implicit method enables to reduce dramatically the memory requirements which makes the method truly interesting to solve viscous flows despite aspect ratio issues.

	Coarse Grid 101×21		Medium Grid 201×41		Fine Grid 301×61	
	Memory storage (MB)	relative size	Memory storage (MB)	relative size	Memory storage (MB)	relative size
Explicit scheme	3.8	1	15.3	1	33.8	1
MF-PJ scheme	5.8	1.53	23.7	1.55	52.6	1.56
MF-SGS scheme	6	1.58	24.5	1.6	54.3	1.61
BL-ALGS scheme	8.4	2.21	32.2	2.10	71.5	2.11

Table 8.4.3: Memory storage for all the methods with respect to the grid size. The explicit scheme is taken as the reference so as to evaluate the relative sizes.

8.5 Stokes' second problem

8.5.1 Description

Now let us compute the flow over an infinite flat oscillating plate moving at the velocity $\tilde{u}(\tilde{t}) = \tilde{u}_\infty \cos(\omega \tilde{t})$. This motion involves steady oscillations that can be calculated analytically :

$$\tilde{u}(\tilde{y},\tilde{t}) = \tilde{u}_\infty \cdot \exp(-\tilde{y}\sqrt{\frac{\omega}{2\tilde{\nu}}}) \cdot \cos(\omega\tilde{t} - \tilde{y}\sqrt{\frac{\omega}{2\tilde{\nu}}})$$

In order to nondimensionalize the problem, a reference length scale has to be defined. To achieve this, one can use the velocity $\tilde{u}_\infty$ and the pulsation ω :

$$L_\infty = \frac{\tilde{u}_\infty}{\omega}$$

Then using the following Reynolds number definition, $Re = \frac{\tilde{u}_\infty L_\infty}{\tilde{\nu}} = \frac{\tilde{u}_\infty^2}{\tilde{\nu}\ \omega}$, one obtains :

$$u(y,t) = \frac{\tilde{u}(\tilde{y},\tilde{t})}{\tilde{u}_\infty} = \exp(-y\sqrt{\frac{Re}{2}}) \cdot \cos(t - y\sqrt{\frac{Re}{2}})$$

We can define the thickness $\tilde{\delta}$ of the layer as the point where the velocity amplitude has dropped to 1 percent of $\tilde{u}_\infty$. Thus, for the dimensionless problem, we obtain :

$$\exp(-\delta\sqrt{\frac{Re}{2}}) = 0.01 = e^{-4.6}$$

or

$$\delta \approx 3.5 \cdot \sqrt{\frac{Re}{2}}$$

We then choose a squared computational domain such as the effects of the oscillating plate are negligible at the top of the domain. We typically take :

$$L = H \geq 2 \cdot \delta$$

Setting the Reynolds number at 10 leads to a thickness of $\delta = 2$, so we can choose $L = H = 4$. The computational domain is therefore chosen squared and it is presented in figure 8.5.1.

So as to determine the accuracy of each methods, we can compute the skin friction coefficient which characterizes the flow :

$$\mathcal{C}_w = \frac{2 \cdot \tau_w}{\tilde{\rho} \cdot \tilde{u}_\infty^2} = \frac{2 \cdot \tilde{\mu}}{\tilde{\rho} \cdot \tilde{u}_\infty \cdot L_\infty} \cdot \left.\frac{\partial u}{\partial y}\right|_w = \frac{2}{Re} \cdot \left.\frac{\partial u}{\partial y}\right|_w$$

The analytical value of this coefficient is given by :

$$\mathcal{C}_w = \frac{2}{\sqrt{Re}} \cdot \sin(t - \frac{\pi}{4})$$

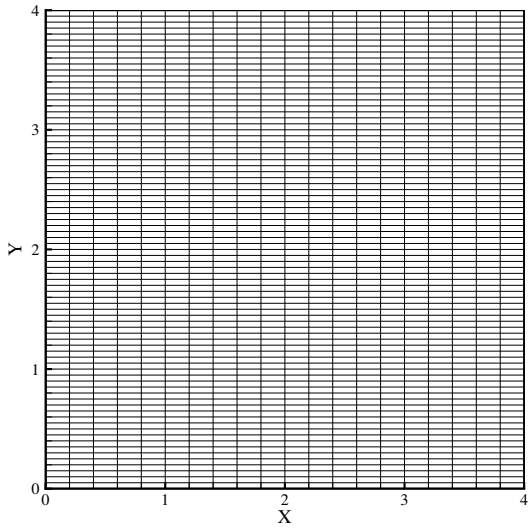


Figure 8.5.1: Computational domain for the Stokes' second problem. The mesh presented here is the coarse one. The oscillating wall is at the bottom.

The accuracy of a numerical solution would be evaluated through the error deviation which is defined as follows :

$$\text{Error} = \sqrt{\frac{1}{2\pi}\sum_{0}^{2\pi}(\mathcal{C}_w - \mathcal{C}_{num})^2}$$

where the numerical skin friction coefficient is calculated using a second-order formula :

$$\mathcal{C}_{num} = \frac{2}{Re} \cdot \frac{-u(j_w+2) + 4u(j_w+1) - 3u(j_w)}{2(y(j_w+1) - y(j_w))}$$

with j_w the value of the space counter at the wall.

Non-dimensional initial values are taken as follows : $M_\infty = 10^{-3}$, $Re = 10$, $\rho_0 = 1$, $u_0 = 0$, $v_0 = 0$, $P_0 = 1/\gamma M_\infty^2$. The no-slip condition is applied at the moving wall while the incompressible conditions are considered for all the others boundaries, that is we impose the static pressure and we extrapolate the density and the velocity. In order to compute the unsteady flows, we use a dual-time-stepping framework, wherein the physical derivatives are employed to follow the physical transients and the pseudo-time derivatives serve as an iterative device. Computations are done using two different grids, namely a coarse one (21×81) and a fine one (41×161).

8.5.2 Objectives of the test-case

In chapter 7, we outlined a new definition of the unsteady low-Mach parameter. We showed that such a definition enables the preconditioned scheme to provide good damping for both acoustic and material problems. In this paragraph, we want to confirm that the unsteady preconditioned scheme is versatile enough for every type of problems. We will study also the performances of the

implicit methods that we have already employed to compute steady flows. The Von Neumann analysis emphasized that Matrix-Free schemes should be very competitive compare to the Block scheme.

8.5.3 Unsteady preconditioning

In chapter 7, we outlined a new definition of the unsteady low-Mach parameter which enables us to avoid the tricky choice of a proper length scale. We recall this new definition :

$$\beta_u = \min\left[\frac{\delta x \cdot \sqrt{CFL_{u+c}}}{\Delta t \cdot c \cdot \pi},\ \frac{\delta y \cdot \sqrt{CFL_{v+c}}}{\Delta t \cdot c \cdot \pi}\right]$$

Then, the low-Mach parameter is classically defined as follows :

$$\beta = \min\left[\max(\beta_s,\ \beta_u),\ 1\right]$$

where β_s is the steady low-Mach parameter. In this paragraph, our objective is to emphasize the good convergence rate provided by the unsteady preconditioned scheme for every physical time-step. So as to study its properties, we consider the MF-SGS(10) scheme with a pseudo CFL number set at 10^6. We compute one physical iteration over the fine grid and we alternatively choose the no-preconditioned scheme, the steady preconditioned scheme and the unsteady preconditioned scheme. The sub-iteration convergence is studied for several time-steps. The time-step can be characterized by $CFL_u = u\Delta t/\delta x$ and, ideally, for efficient time-marching solution, this CFL number should be of order unity.
Figure 8.5.2 shows the results for $M_\infty = 0.001$ and $Re = 10$, and for four values of CFL_u. At such a speed, for low values of CFL_u the steady preconditioner does not converge in the dual iteration. The unsteady preconditioner outperforms the non-preconditioned case even for $CFL_u = 10^{-3}$. At $CFL_u = 1$, the non-preconditioned case diverges while the steady preconditioner provides a quite good convergence rate. However, the unsteady preconditioner is still the most efficient. At even higher $CFL_u = 10$, the steady preconditioner performs as well as its unsteady counterpart. It is thus clear that in all cases the unsteady preconditioning provides optimal performance of the pseudo-time convergence. These results are in good agreement with the Von Neumann analyses of chapter 7.
The dual-iteration convergence is important because it drives the accuracy of the solution : a not-converged process would lead to an inaccurate solution. In practice, it is not necessary to reach machine zero at each physical time-step, we therefore choose a certain level of residual to reach so as to ensure efficiency and accuracy. Of course, accuracy relies also on the numerical dissipation, hence, a no-preconditioned scheme would generate excessive artificial dissipation leading to a wrong solution. Such an issue can also occur when one uses the unsteady preconditioning with a small time-step; indeed, in this configuration, the unsteady preconditioning is close to the no-preconditioned case and it can therefore provide an inaccurate solution. However, for the problem that we are interested in, the time-steps correspond to $CFL_u \approx 1$, thus the unsteady preconditioned scheme is expected to be as accurate as the steady preconditioned one. We computed the steady oscillations during 4 periods over the coarse grid for each method. We used a time-step equal to $\pi/20$ which corresponds to $CFL_u \approx \pi/4$. Figure 8.5.3 shows the evolution of the skin friction coefficient for the last period computed and the error made by each method over the same period. We obtain the same accuracy with both steady and unsteady preconditioning techniques : the error committed by each scheme is of the order of the percent on the coarse

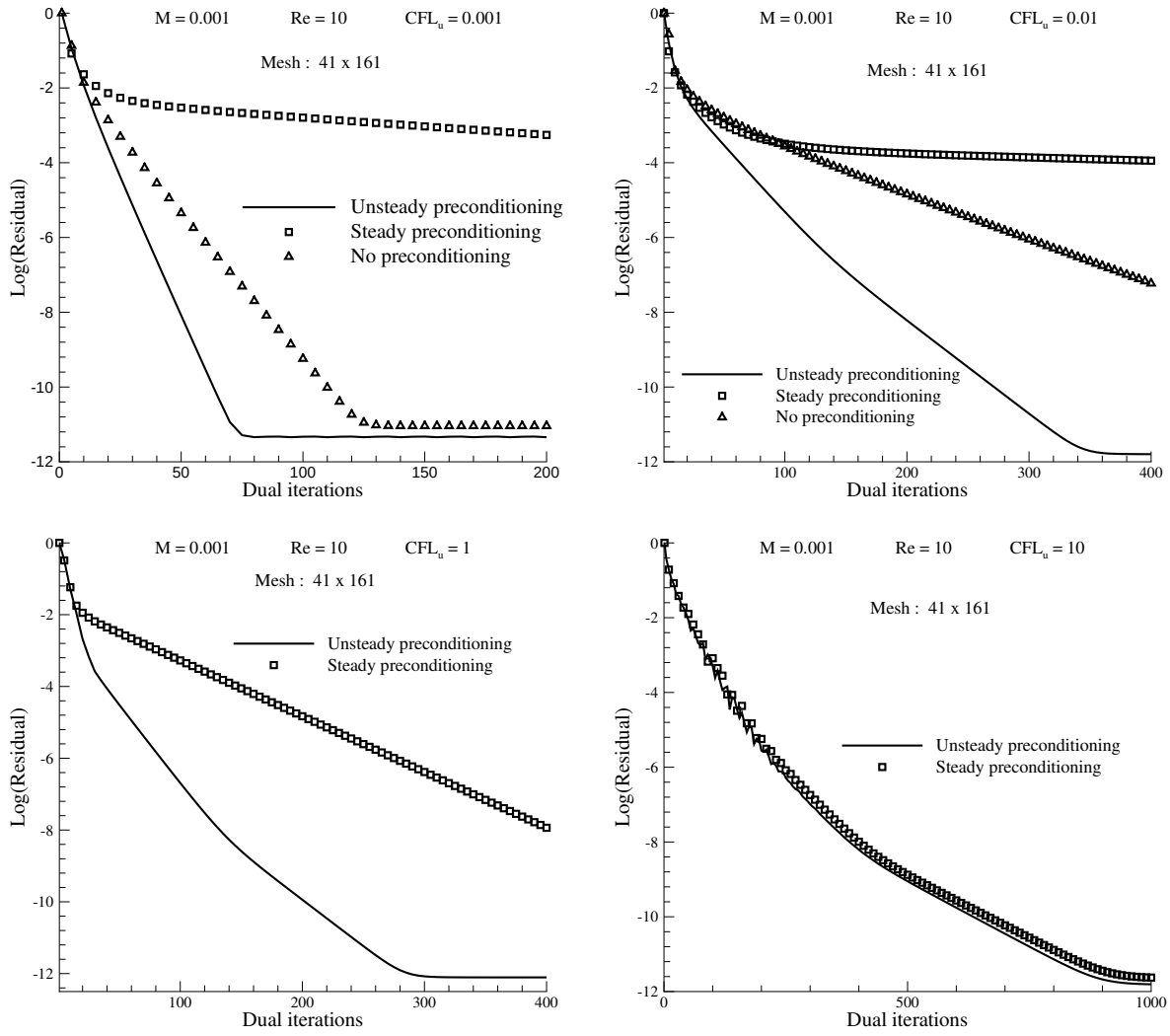
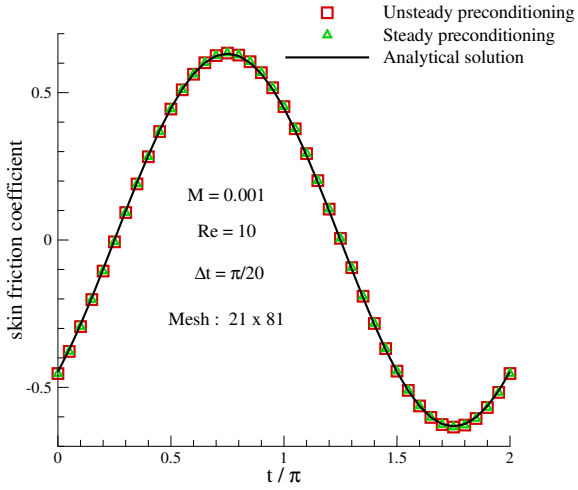


Figure 8.5.2: Comparisons of residual convergence with and without unsteady preconditioning at $M_\infty = 10^{-3}$ and $Re = 10$ and on the fine grid (41×161). Top-Left : $CFL_u = 10^{-3}$. Top-Right : $CFL_u = 10^{-2}$. Bottom-Left : $CFL_u = 1$. Bottom-Right : $CFL_u = 10$.

grid. The unsteady preconditioning enables to scale properly the artificial dissipation for such a case. Besides, as it provides very fast convergence we will employ it for the next comparisons between Matrix-Free method and Block scheme.

8.5.4 Comparisons Block scheme / Matrix-Free scheme

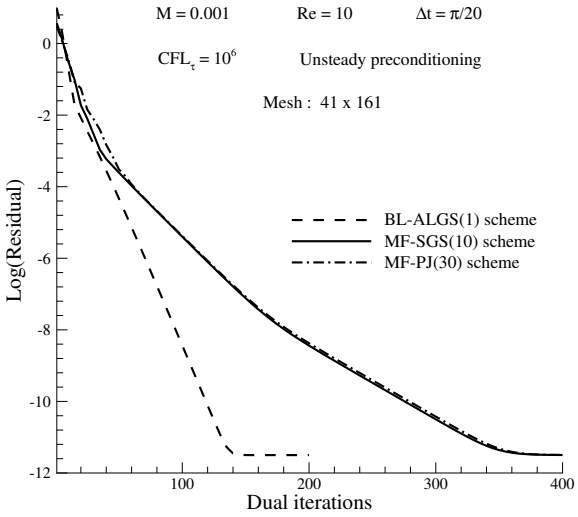
As we mentioned earlier, practical computations do not require to reach machine zero in pseudo-convergence at each physical time-step. It is customary to target a certain order of convergence before leaving the dual-time loop in the program. In this section, we want to emphasize that the choice of the order of convergence is the main parameter which drives the efficiency of our Matrix-Free implicit schemes. In figure 8.5.4, we present the dual-time convergence history of



Preconditioning	Error
Steady	$1.17 \cdot 10^{-2}$
Unsteady	$1.17 \cdot 10^{-2}$

Figure 8.5.3: Left : Skin friction coefficient for the last period computed. $M_\infty = 10^{-3}$, $Re = 10$, $\Delta t = \pi/20$ and mesh 21×81. Right : Table summarizing the error for each method.

the residual after one physical iteration for BL-ALGS(1) scheme, MF-SGS(10) scheme and MF-PJ(30) scheme at $M_\infty = 10^{-3}$, $Re = 10$, $\Delta t = \pi/20$ and on the fine grid (41×161).



Residual	Method	Dual Iterations
8	BL-ALGS(1)	**95**
	MF-SGS(10)	180
	MF-PJ(30)	180
4	BL-ALGS(1)	**45**
	MF-SGS(10)	60
	MF-PJ(30)	60

Figure 8.5.4: Left : Dual-time convergence history of the residual after one physical iteration at $M_\infty = 10^{-3}$, $Re = 10$, $\Delta t = \pi/20$ and on the fine grid. Right : Table showing the amount of dual-iterations required to reach a certain order of convergence for each method.

We observe that Matrix-Free schemes require about 2.7 times as many iterations as the Block scheme to reach machine zero. On the other hand, this figure decreases to 2 to reach 8-order convergence and about 4/3 to reach 4-order convergence. Hence, according to the order of convergence which is targeted, the performance of the Matrix Free schemes can be very different. To emphasize that, we compute successively four periods of the steady oscillations for different

levels of convergence. In Table 8.5.1, we present the results obtained when 8-order convergence is required. The Block scheme provides of course the best intrinsic efficiency, however thanks to their low-cost per iteration the Matrix-Free schemes, especially the MF-SGS one, remain competitive. On the other hand, when only 4-order convergence is required, the Matrix-Free schemes become very efficient in terms of iterations and CPU time. Besides, we notice that the accuracy does not deteriorate. In fact, accuracy is mainly driven by the time-step size rather than the order of convergence. Of course, a too small order of convergence would lead to inaccurate solution. In practice, 4 or 5-order convergence is required to ensure accuracy. The Matrix-Free implicit method is therefore very interesting to compute unsteady low-Mach flows.

Mesh	Method	CFL	Dual Iterations	CPU	CPI	CPPPI	Error
21×81	BL-ALGS(1)	10^6	**7130**	981 s.	0.138	$8.1 \cdot 10^{-5}$	$1.17 \cdot 10^{-2}$
	MF-SGS(10)	10^6	16305	**905 s.**	0.056	**3.3** $\cdot 10^{-5}$	
	MF-PJ(30)	10^6	16545	1106 s.	0.067	$3.9 \cdot 10^{-5}$	
41×161	BL-ALGS(1)	10^6	**13700**	8265 s.	0.6	$9.1 \cdot 10^{-5}$	$5.8 \cdot 10^{-3}$
	MF-SGS(10)	10^6	25465	**7632 s.**	0.3	**4.5** $\cdot 10^{-5}$	
	MF-PJ(30)	10^6	26425	9572 s.	0.36	$5.5 \cdot 10^{-5}$	

Table 8.5.1: Viscous flow over an oscillating flat plate at $M_\infty = 10^{-3}$ and $Re = 10$. The physical time-step is equal to $\pi/20$ in every case while the pseudo-CFL number is set at 10^6. The order of convergence required for each physical time-step is set at 8.

8.5.5 Conclusions

In this test-case, we studied the modification of the unsteady preconditioning selection that we outlined in chapter 7. Such a modification significantly enhances the capability of the dual-time scheme, enabling it to handle unsteady flows over a wide range of Mach numbers and time scales. Specifically, the new definition of the unsteady low-Mach parameter improves efficiency by optimizing the number of dual-iterations required at each physical time-step without defining a specific length scale. Besides, the preconditioned dual-time Matrix-Free schemes were shown to be very competitive with respect to classical Block formulation. When the required order of convergence is not too large, the Matrix-Free method clearly outperforms the Block algorithm. Furthermore, the reduction of memory requirements that we emphasized for steady flows is still valid, making our Matrix-Free formulation very interesting to compute unsteady flows.

Mesh	Method	CFL	Dual Iterations	CPU	CPI	CPPPI	Error
21×81	BL-ALGS(1)	10^6	**3315**	460 s.	0.138	$8.1 \cdot 10^{-5}$	$1.17 \cdot 10^{-2}$
	MF-SGS(10)	10^6	4795	**268 s.**	0.056	**3.3** $\cdot 10^{-5}$	
	MF-PJ(30)	10^6	5000	335 s.	0.067	$3.9 \cdot 10^{-5}$	
41×161	BL-ALGS(1)	10^6	**5320**	3075 s.	0.58	$8.8 \cdot 10^{-5}$	$5.8 \cdot 10^{-3}$
	MF-SGS(10)	10^6	7665	**2300 s.**	0.31	**4.5** $\cdot 10^{-5}$	
	MF-PJ(30)	10^6	8885	3229 s.	0.36	$5.5 \cdot 10^{-5}$	

Table 8.5.2: Viscous flow over an oscillating flat plate at $M_\infty = 10^{-3}$ and $Re = 10$. The physical time-step is equal to $\pi/20$ in every case while the pseudo-CFL number is set at 10^6. The order of convergence required for each physical time-step is set at 4.

Part III

A Matrix-Free Implicit Treatment for the CAST3M Code

Chapter 9

Matrix-Free Implicit Strategy for Unstructured Grids

9.1 Presentation of the CAST3M code

During certain postulated accidents, hydrogen may be released or generated into nuclear reactor containments, tokamak vacuum vessels and more generally into industrial units where hydrogen risk analysis has to be taken into account. As combustible quantities of hydrogen, air and steam may form locally, it is necessary to predict accurately the distribution of the different gases within the geometry in order to simulate the combustion of the hydrogen. Indeed, depending on the local concentration and/or the presence and activation of mitigation devices, hydrogen may burn following different modes (diffusion flames, local or global deflagration, accelerated flames, detonation), or may not burn at all. However, the simulation of H_2 distribution and combustion in such geometries is a challenging task from the point of view of numerical simulation, as it involves quite disparate length and times scales, which need to be resolved appropriately and efficiently. The French Atomic Energy Commission (CEA) is involved in the development and validation of codes to model such problems, for external clients such as IRSN, TECHNICATOME, or for its own safety studies. All these codes are implemented and developed in the same computational platform, the CAST3M code.
The CAST3M code incorporates both lumped-parameter and multi-dimensional formulations. The lumped-parameter approach enables to represent the main flow of the mixture and the compartment average distribution for the different mixture species with a relatively small number of compartments and junctions between them. Hence, the simulation can cover several hours of physical time at a relatively low CPU cost. However, due to the limitations of the approach, multi-dimensional effects such as local heat transfer cannot be modeled. Multi-dimensional approach has to deal with a wide range of spatial and time scales. The release and mixing of gases in case of stratification is a long process, while the combustion phase is a short-lived phenomenon. Furthermore, the burning modes of H_2 range from nearly incompressible (deflagration) to highly compressible (detonation).
A possible option to cope with this challenging simulations is to develop for each type of flow the most suitable numerical method. Thus, the CAST3M code features a pressure-based solver so as to carry out distribution calculations and a density-based solver in order to solve combustion calculation. The pressure-based approach relies on an asymptotic approximation of the

Navier-Stokes equations in the limit of small Mach numbers and leads to elliptic models in which the acoustic waves have been filtered out. Moreover, the spatial discretization of the equations is obtained by a Finite Element method while the time discretization is obtained through an incremental implicit second order projection method. On the other hand, the density-based solver uses shock-capturing conservative methods featuring different schemes such as Roe, Van Leer or AUSM. It is applied in a unstructured cell-centered Finite Volume framework for the spatial discretization. Explicit and implicit time-integrations are available, and in the latter case a Newton-Krylov strategy was first used.
An alternative option to face with the different scales is to develop a single method suitable for all flows regimes. Compressible flow solvers modified through the use of "preconditioning" techniques enables to preserve efficiency and accuracy even when the Mach number becomes small. The preconditioning matrix rescales properly the numerical dissipation and improves the condition number of the system. Turkel-type preconditioning for Flux Difference splitting schemes [75] and a low Mach number version of the AUSM+ scheme [25] have been implemented. Then, the system is solved with the implicit Newton-Krylov technique. This method enables the use of large time-steps to compute low-Mach number flow but it requires also the storage of Jacobian matrices, which can become prohibitive when dealing with problems with a high number of degrees of freedom.
The Matrix-Free implicit framework that we presented in the previous chapters aims precisely at reducing the memory overhead by simplifying the resolution of the system. In this chapter, our objective is to extend the Matrix-Free implicit method to the unstructured Finite Volume formula. We will see that such an extension can be carried out straightforwardly. Next chapters will be dedicated to the study of the efficiency of the Matrix-Free technique with respect to the existent methods in CAST3M.

9.2 Numerical methods for unstructured grids

A time-accurate solution of the Navier-Stokes equations is computed for flows at all speeds by solving :

$$P^{-1}\frac{\partial w}{\partial \tau} + \frac{\partial w}{\partial t} + \frac{1}{\Omega_i}\sum_k (f^E_{i,k} - f^V_{i,k}) \cdot \vec{n}_{i,k} S_{i,k} = 0 \tag{9.2.1}$$

where w is the vector of the conserved variable, τ is the pseudo or dual-time, t is the physical time, Ω_i is the volume of the i-th cell (*cf.* figure 9.2.1), $f^E_{i,k}$ and $f^V_{i,k}$ are respectively the convective and viscous fluxes at the face k, $\vec{n}_{i,k}$ is the outward normal and $S_{i,k}$ is the surface of the face k. Eventually, P is the low-Mach preconditioning matrix presented earlier.
System (9.2.1) is solved using the numerical scheme :

$$(P^{-1})^{n,m}_i \frac{\Delta w^{n,m}_i}{\Delta \tau^{n,m}_i} + \frac{\frac{3}{2}(w^{n,m+1}_i - w^n_i) - \frac{1}{2}\Delta w^{n-1}_i}{\Delta t} + \frac{1}{\Omega_i}\sum_k \left(\mathcal{F}^E_{i,k} - \mathcal{F}^V_{i,k}\right)^{n,m+1} \cdot \vec{n}_{i,k}\, S_{i,k} = 0 \tag{9.2.2}$$

where m is the pseudo-iteration counter, n is the time step counter, $\Delta w^{n,m} = w^{n,m+1} - w^{n,m}$, $\Delta w^{n-1} = w^n - w^{n-1}$ and $\mathcal{F}^E_{i,k}$ [1] (respectively $\mathcal{F}^V_{i,k}$) is the numerical approximation of the flux

[1] Only preconditioned numerical inviscid fluxes are used in this chapter but for the sake of clarity we will omit the superscript $\tilde{}$.

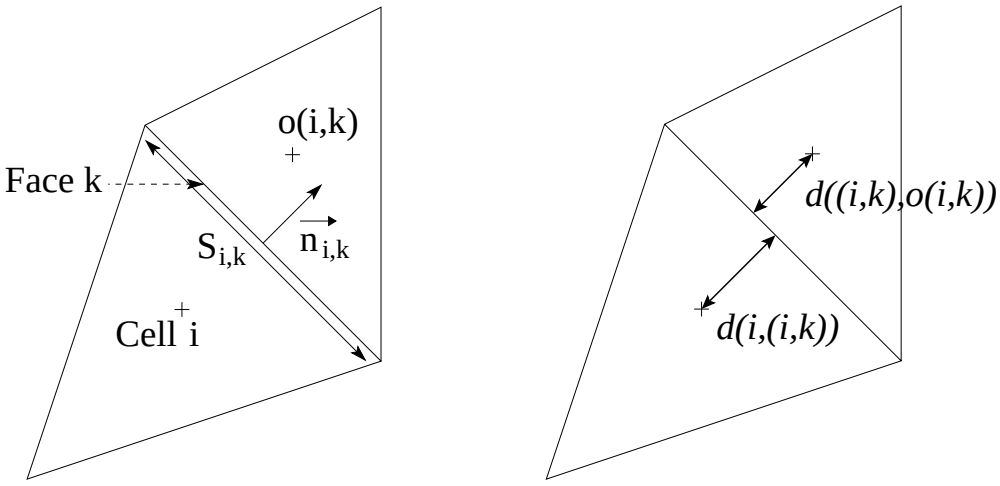


Figure 9.2.1: Control cell for unstructured grid.

vector f^E (respectively f^V). In the manner of the Cartesian case, the linearization of the numerical fluxes around pseudo-time level (n, m) leads to :

$$(P_c^{-1})_i^{n,m} \frac{\Delta w_i^{n,m}}{\Delta \tau_i^{n,m}} + \frac{3}{2} \frac{\Delta w_i^{n,m}}{\Delta t} + \frac{1}{\Omega_i} \sum_k \Delta \mathcal{F}_{i,k}^{n,m} \cdot \vec{n}_{i,k} \; S_{i,k} = -\mathcal{R}_i^{n,m} \; , \tag{9.2.3}$$

where the explicit residuum is given by :

$$\mathcal{R}_i^{n,m} = \frac{1}{\Omega_i} \sum_k \mathcal{F}_{i,k}^{n,m} \cdot \vec{n}_{i,k} \; S_{i,k} + \frac{\frac{3}{2}(w_i^{n,m} - w_i^n) - \frac{1}{2}\Delta w_i^{n-1}}{\Delta t} \tag{9.2.4}$$

and $\mathcal{F} = \mathcal{F}^E - \mathcal{F}^V$. As far as the RHS of (9.2.3) is concerned, inviscid fluxes are computed using classical upwind schemes (Roe, AUSM+), extended to higher order using the primitive variable reconstruction of Barth and Jespersen, described in [2] and belonging to the MUSCL family; besides the numerical dissipation of the inviscid schemes is corrected by taking into account Low Mach number preconditioning. Viscous fluxes are approximated using a linearly-exact extension of the diamond method of Noh [60]. On the other hand, the inviscid fluxes of the LHS of (9.2.3) are evaluated using first-order space accuracy. Such a choice has no effect on the accuracy of the solution since at convergence we have :

$$\lim_{m \to \infty} \Delta w^{n,m} = 0$$

and, as a result, the LHS of equation (9.2.3) goes to zero. For a large class of schemes, we then have :

$$\mathcal{F}_{i,k}^E \cdot \vec{n}_{i,k} = \frac{1}{2} \left(f_i^E \cdot \vec{n}_{i,k} + f_{o(i,k)}^E \cdot \vec{n}_{i,k} \right) + \frac{1}{2} Q_{i,k}^E \left(w_i - w_{o(i,k)} \right) \; , \tag{9.2.5}$$

where Q^E is the (positive-definite) numerical diffusion matrix and the index $o(i,k)$ refers to the element which shares with the i-th element its k-th interface ("o" standing for "opposite"

element (see also figure 9.2.1)). The time increment of the numerical inviscid flux is then given by :

$$\begin{aligned}\left(\Delta\mathcal{F}^E_{i,k}\right)^{n,m}\cdot\vec{n}_{i,k} &= \frac{1}{2}\left(\left(\Delta f^E_i\right)^{n,m}-\left(\Delta f^E_{o(i,k)}\right)^{n,m}\right)\cdot\vec{n}_{i,k}+\frac{1}{2}\Delta\left(Q^E_{i,k}(w_i-w_{o(i,k)})\right)^{n,m}\\ &\approx \frac{1}{2}\left(\left(\Delta f^E_i\right)^{n,m}-\left(\Delta f^E_{o(i,k)}\right)^{n,m}\right)\cdot\vec{n}_{i,k}+\frac{1}{2}\left(Q^E_{i,k}\right)^{n,m}\Delta\left(w_i-w_{o(i,k)}\right)^{n,m}\end{aligned} \tag{9.2.6}$$

Since

$$\sum_k\left(\Delta\mathcal{F}^E_i\right)^{n,m}\cdot\vec{n}_{i,k}S_{i,k}=\left(\Delta\mathcal{F}^E_i\right)^{n,m}\cdot\left(\sum_k\vec{n}_{i,k}S_{i,k}\right)=0\ ,$$

it follows that, using formula (9.2.6),

$$\begin{aligned}\sum_k\left(\Delta\mathcal{F}^E_{i,k}\right)^{n,m}\cdot\vec{n}_{i,k}S_{i,k} &= \frac{1}{2}\left(\sum_k\left(\Delta f^E_{o(i,k)}\right)^{n,m}\cdot\vec{n}_{i,k}S_{i,k}\right)\\ &\quad+\frac{1}{2}\left(\sum_k\left(Q^E_{i,k}\right)^{n,m}\Delta\left(w_i-w_{o(i,k)}\right)^{n,m}S_{i,k}\ .\right)\end{aligned} \tag{9.2.7}$$

As far as the evaluation of the diffusive flux in the LHS of (9.2.3) is concerned, we neglect the dependence of the viscous flux with respect to the derivative of the variables in the tangential direction ($\vec{t}$). Thus, we have :

$$\mathcal{F}^V_{i,k}\cdot\vec{n}_{i,k}=A_{\vec{n},i,k}\cdot\left(\frac{\partial w}{\partial\vec{n}}\right)_{i,k}+A_{\vec{t},i,k}\cdot\left(\frac{\partial w}{\partial\vec{t}}\right)_{i,k}\approx A_{\vec{n},i,k}\cdot\left(\frac{\partial w}{\partial\vec{n}}\right)_{i,k}$$

Then, we evaluate the normal derivative of w in the n-direction as follows :

$$\left(\frac{\partial w}{\partial\vec{n}}\right)_{i,k}\approx\frac{w_{o(i,k)}-w_i}{d(i,(i,k))+d((i,k),o(i,k))}$$

where $(d(i,(i,k))+d((i,k),o(i,k)))$ is the sum of the distances between the involved cell-centers and the interface (*cf.* figure 9.2.1). Eventually, we obtain the following expression :

$$\mathcal{F}^V_{i,k}\cdot\vec{n}_{i,k}=Q^V_{i,k}\cdot\left(w_{o(i,k)}-w_i\right) \tag{9.2.8}$$

where the matrix $Q^V_{i,k}$ is defined as follows :

$$Q^V_{i,k}=\frac{1}{d(i,(i,k))+d((i,k),o(i,k))}\cdot A_{\vec{n},i,k}\ .$$

Using formula (9.2.8) and neglecting the variation of $Q^V_{i,k}$ from m to $m+1$ yields :

$$\sum_k\left(\Delta\mathcal{F}^V_{i,k}\right)^{n,m}\cdot\vec{n}_{i,k}S_{i,k}\ =\ -\left(\sum_k\left(Q^V_{i,k}\right)^{n,m}\Delta\left(w_i-w_{o(i,k)}\right)^{n,m}S_{i,k}\right)\ . \tag{9.2.9}$$

Finally, inserting (9.2.7) and (9.2.9) into the LHS of equation (9.2.3), we find the implicit scheme for unstructured grids :

$$\begin{aligned} D_i^{n,m} \cdot \Delta w_i^{n,m} + \left(\frac{1}{2\Omega_i} \sum_k \left(\Delta f^E_{o(i,k)} \right)^{n,m} \cdot \vec{n}_{i,k} \; S_{i,k} \right) \\ - \left(\frac{1}{\Omega_i} \sum_k \left(Q^V + \frac{1}{2} Q^E \right)^{n,m}_{i,k} \cdot \Delta w^{n,m}_{o(i,k)} \; S_{i,k} \right) = -\mathcal{R}_i^{n,m} \end{aligned} \tag{9.2.10}$$

with

$$D_i^{n,m} \; = \; \left(\frac{1}{\Delta\tau} P^{-1} \right)^{n,m}_i \; + \; \frac{3}{2\Delta t} Id \; + \; \left(\frac{1}{\Omega_i} \sum_k \left(Q^V + \frac{1}{2} Q^E \right)^{n,m}_{i,k} \; S_{i,k} \right)$$

In the particular case of the Cartesian grids, one recover the numerical scheme detailed in the first part of the thesis.

9.3 Matrix-Free implicit treatment

The Matrix-Free framework can be straightforwardly applied to the unstructured scheme (9.2.10). First, we proceed to the spectral radius simplification on the dissipative terms of the LHS :

$$Q^E_{i,k} \approx \rho(PA^E)_{i,k} \cdot P^{-1}_{i,k} = \tilde{\rho}^E_{i,k} \cdot P^{-1}_{i,k}$$

$$Q^V_{i,k} \approx \rho(Q^V_{i,k}) \cdot Id = \rho^V_{i,k} \cdot Id$$

The unstructured scheme then becomes :

$$D_i^{n,m} \cdot \Delta w_i^{n,m} + \left(\frac{1}{2\Omega_i} \sum_k \left(\Delta f^E_{o(i,k)} \right)^{n,m} \cdot \vec{n}_{i,k} \; S_{i,k} \right) - \left(\frac{1}{\Omega_i} \sum_k C^{n,m}_{i,k} \cdot \Delta w^{n,m}_{o(i,k)} \; S_{i,k} \right) = -\mathcal{R}_i^{n,m}$$

with

$$\begin{cases} D_i^{n,m} \; = \; \left(\frac{1}{\Delta\tau} P^{-1} \right)^{n,m}_i \; + \; \frac{3}{2\Delta t} Id \; + \; \left(\frac{1}{\Omega_i} \sum_k \left(\rho^V Id + \frac{1}{2} \tilde{\rho}^E P^{-1} \right)^{n,m}_{i,k} \; S_{i,k} \right) , \\ C^{n,m}_{i,k} = \left(\rho^V Id + \frac{1}{2} \tilde{\rho}^E P^{-1} \right)^{n,m}_{i,k} . \end{cases}$$

In order to produce a Matrix-Free implicit method, all the non-diagonal terms are relaxed in the manner of the Point Jacobi procedure to yield :

$$D_i^{n,m} \cdot \Delta w_i^{(l+1)} = -\mathcal{R}_i^{n,m} - \left(\frac{1}{2\Omega_i} \sum_k \left(\Delta f^E_{o(i,k)} \right)^{(l)} \cdot \vec{n}_{i,k} \; S_{i,k} \right) + \left(\frac{1}{\Omega_i} \sum_k C^{n,m}_{i,k} \cdot \Delta w^{(l)}_{o(i,k)} \; S_{i,k} \right) \tag{9.3.1}$$

In the particular case of the compressible flows, in which the preconditioning is not required, the preconditioner P is equal to the identity matrix so that the coefficients D and C reduce to :

$$\begin{cases} D_i^{n,m} = (a_i^{n,m} + b_i^{n,m})\, Id \ , \\ \\ C_{i,k}^{n,m} = \left(\rho^V + \dfrac{1}{2}\rho^E\right)_{i,k}^{n,m} \ , \end{cases}$$

with

$$a_i^{n,m} = \left(\frac{1}{\Delta\tau_i^{n,m}} + \frac{1}{2\Omega_i}\sum_k (\rho_{i,k}^E)^{n,m}\ S_{i,k}\right) \quad \text{and} \quad b_i^{n,m} = \left(\frac{3}{2\Delta t} + \frac{1}{\Omega_i}\sum_k (\rho_{i,k}^V)^{n,m}\ S_{i,k}\right) \ .$$

Thus, the unstructured scheme (9.3.1) is truly Matrix-Free and it does not require neither matrix inversion nor matrix storage. However, when we compute low-Mach number flows, the preconditioning matrix differs from the identity. Consequently, the diagonal coefficient D becomes a full matrix to be inverted, and C are also block coefficients, so that storage requirement and operations count of the implicit treatment significantly increase. Nevertheless, as shown in the Cartesian case, it is possible to take advantage of the properties of P to keep the preconditioned approach as simple as its compressible flow version.

In the first chapter, the Turkel preconditioner written in conservative variables was presented :

$$P = Id + (\beta^2 - 1)\cdot Q$$

with

$$Q = \frac{\gamma-1}{c^2}\begin{bmatrix} q^2 = \frac{1}{2}(u^2+v^2) & -u & -v & 1 \\ uq^2 & -u^2 & -uv & u \\ vq^2 & -uv & -v^2 & v \\ Hq^2 & -uH & -vH & H \end{bmatrix} = \frac{\gamma-1}{c^2}\left[\begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix} \cdot \left(q^2\ ;\ -u\ ;\ -v\ ;\ 1\right)\right]$$

The matrix Q is idempotent, *i.e.* $Q^2 = Q$, hence, the preconditioning matrix can be inverted explicitly :

$$P^{-1} = Id + (\frac{1}{\beta^2} - 1)\cdot Q$$

In order to exploit this property, in the matrix $D_i^{n,m}$, we evaluate all the matrices P at the center of the i-th cell, thus, we obtain :

$$D_i^{n,m} = a_i^{n,m}\left(P^{-1}\right)_i^{n,m} + b_i^{n,m} Id$$

Then, the matrix is inverted in the same way of the Cartesian case :

$$(D_i^{n,m})^{-1} = \left(\frac{1}{a+b}\left(Id + \frac{a\left(\beta^2-1\right)}{a+b\beta^2}Q\right)\right)_i^{n,m}$$

Besides, in coefficient C, we can also evaluate the matrix P at the center of the i-th cell which leads to the following expression :

$$\begin{aligned}\Delta w_i^{(l+1)} = (D_i^{n,m})^{-1} \cdot \left[-\mathcal{R}_i^{n,m} - \frac{1}{\Omega_i} \sum_k \left\{ \frac{1}{2} \left(\Delta f_{o(i,k)}^E \right)^{(l)} \cdot \vec{n}_{i,k} - (\rho_{i,k}^V)^{n,m} \cdot \Delta w_{o(i,k)}^{(l)} \right\} S_{i,k} \right] \\ + (D_i^{n,m})^{-1} \cdot (P_i^{n,m})^{-1} \cdot \left[\frac{1}{2\Omega_i} \sum_k \tilde{\rho}_{i,k}^E \cdot \Delta w_{o(i,k)}^{(l)} \cdot S_{i,k} \right]\end{aligned} \tag{9.3.2}$$

As shown in chapter 4, the product $D^{-1} \cdot P^{-1}$ is explicitly given by :

$$(D_i^{n,m})^{-1} \cdot (P_i^{n,m})^{-1} = \left(\frac{1}{a+b} \left(Id + \frac{b\left(\beta^2 - 1\right)}{a + b\beta^2} Q \right) \right)_i^{n,m}$$

Hence, the expression (9.3.2) reads :

$$\begin{aligned}\Delta w_i^{(l+1)} = & \frac{1}{(a+b)_i^{n,m}} \left[\Delta w_1^{(l)} + \Delta w_2^{(l)} \right]_i \\ & + \frac{(\beta^2)_i^{n,m} - 1}{((a+b)(a+b\beta^2))_i^{n,m}} \; Q_i^{n,m} \left[a\Delta w_1^{(l)} + b\Delta w_2^{(l)} \right]_i\end{aligned} \tag{9.3.3}$$

with

$$\begin{cases} \left(\Delta w_1^{(l)} \right)_i = -\mathcal{R}_i^{n,m} - \dfrac{1}{\Omega_i} \displaystyle\sum_k \left\{ \frac{1}{2} \left(\Delta f_{o(i,k)}^E \right)^{(l)} \cdot \vec{n}_{i,k} - (\rho_{i,k}^V)^{n,m} \cdot \Delta w_{o(i,k)}^{(l)} \right\} S_{i,k} \\ \left(\Delta w_2^{(l)} \right)_i = \dfrac{1}{2\Omega_i} \displaystyle\sum_k \tilde{\rho}_{i,k}^E \cdot \Delta w_{o(i,k)}^{(l)} \cdot S_{i,k} \end{cases}$$

This implicit treatment enables to decouple nicely the standard Matrix-Free implicit treatment used for compressible flow ($\beta = 1$) from the added treatment specific to low-Mach number flows ($\beta \neq 1$). Moreover, in the manner of the chapter 4, the treatment (9.3.3) becomes wholly Matrix-Free if the specific expression of matrix Q is employed to compute a matrix-vector product such as $Q \cdot X$ under the form :

$$Q \cdot X = \frac{\gamma - 1}{c^2} \left(q^2 X^{(1)} - u X^{(2)} - v X^{(3)} + X^{(4)} \right) \cdot \begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix}$$

with $X^{(m)}$ the components of vector X. Eventually, $\Delta w_i^{(l+1)}$ can be computed at a low cost and with a reduced memory storage using the following expression :

$$\Delta w_i^{(l+1)} = \frac{1}{(a+b)_i^{n,m}} \left[\Delta w_1^{(l)} + \Delta w_2^{(l)} \right]_i + \Delta \phi_i^{(l)} \cdot \begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix}_i^{n,m} \tag{9.3.4}$$

where the increment $\Delta\phi_i^{(l)}$ is given by :

$$\Delta\phi_i^{(l)} = \chi_i^{n,m} \left[q^2 \left(a\Delta w_1^{(1)} + b\Delta w_2^{(1)} \right) - u \left(a\Delta w_1^{(2)} + b\Delta w_2^{(2)} \right) \right.$$
$$\left. -v \left(a\Delta w_1^{(3)} + b\Delta w_2^{(3)} \right) + \left(a\Delta w_1^{(4)} + b\Delta w_2^{(4)} \right) \right]_i^{(l)}$$

with

$$\chi_i^{n,m} = \left[\frac{\gamma - 1}{c^2} \cdot \frac{(\beta^2) - 1}{(a+b)(a+b\beta^2)} \right]_i^{n,m} .$$

9.4 Conclusion

Thus, the Matrix-Free implicit framework extends easily to the unstructured grid formulation. The form of the final algorithm remains the same than for the Cartesian case. Let us notice that the previous algorithm can be coupled also with the Symmetric Gauss-Seidel relaxation procedure, which is expected to improve the convergence rate of the method. Next chapters are dedicated to the validation of the Matrix-Free technique that have been implemented in the CAST3M code. We will compare it with respect to the Newton-Krylov algorithm and we will also carry out comparisons with the asymptotic pressure-based solvers of CAST3M. s

Chapter 10

Validation of the Matrix-Free Method for 2D-3D Euler Test-Cases

10.1 Introduction

The Newton-Krylov approach, which has been firstly implemented in the CAST3M code, theoretically allows to compute stationary solutions without time steps restriction linked to the CFL condition. However, it requires computing and storing Jacobian matrices and, consequently, when meshes with a huge number of elements are used, memory storage becomes an important issue. The Matrix-Free implicit treatment which was presented in the previous chapters has been developed to reduce the memory storage and it is also hoped that by simplifying the resolution, gains in CPU can also be achieved. This Chapter is devoted to the validation of the newly implemented method in CAST3M over 2D-3D Euler test-cases. The promising properties of the MF schemes which were exhibited in the Cartesian tests are expected to be preserved over unstructured grids.

10.2 Flow over a sine bump

10.2.1 Position of the problem

An inviscid flow in a channel is computed. The geometry of the channel consists of a rectangular domain $(x, y) \in [0; 4] \times [0; 1]$, with a straight upper wall and a curved lower wall, whose coordinates are given by :

$$\begin{cases} x < 1 & y = 0 \\ 1 \geq x \geq 3 & y = 0.1\ (1 - \cos[(x-1)\pi]) \\ x > 3 & y = 0 \end{cases}$$

Four irregular grids are employed to carry out the studies :

- a coarse grid with 878 triangular elements (*cf.* figure 10.2.1);
- a medium grid with 3468 triangular elements;

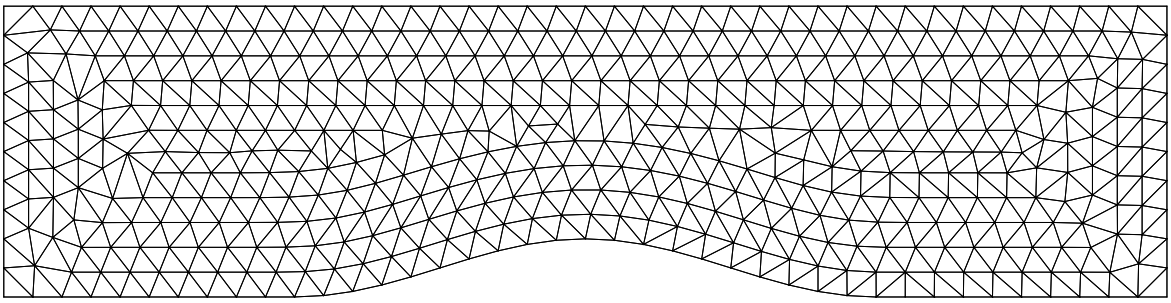


Figure 10.2.1: Stationary flow over a sine bump. Irregular coarse mesh with 878 triangular elements.

- a fine grid with 7898 triangular elements;
- a very fine grid with 14104 triangular elements.

For each grid, the Euler equations are solved for an ideal gas, with $\gamma = 1.4$. For $M_\infty \leq 0.5$ the flow remains subsonic, and hence isentropic and irrotational (because for higher Mach numbers one gets a transonic flow with a shock), and the Mach isolines are expected to be completely symmetrical with respect to the geometry (*i.e.* potential flow). Non-dimensional values are taken as follows : $\rho_0 = 1.4$, $u_0 = M_\infty$, $v_0 = 0$, $P_0 = 1$. Subsonic boundary conditions are considered :

- at the inlet (left) we impose the external total enthalpy, the external entropy and the flow angle while we extrapolate the pressure from inside;
- at the outlet (right) we impose the static pressure while we extrapolate the density and the velocity from inside;
- at the top and at the bottom, we impose wall boundary conditions.

10.2.2 Purpose

Three different values of the inlet Mach number are considered, namely $M_\infty = 0.1$, $M_\infty = 10^{-4}$ and $M = 0.5$. Our purpose is to evaluate in terms of CPU time the following approaches :

- the Point Jacobi Matrix-Free method with α inner-iterations that we will denote MF-PJ(α);
- the Symmetric Gauss-Seidel Matrix-Free method with α inner-iterations that we will denote MF-SGS(α);
- the Newton-Krylov algorithm (GMRES with ILUT preconditioner, Krylov space dimension equal to 50) for which we consider that the convergence of the iterative scheme for the linear system is achieved once the linear residuum is lower than $10^{-\alpha}$ (NK(α)).

We directly solve the stationary Euler equations and we use the AUSM+(P) scheme introduced by Edwards & Liou [25] and coupled with a non-limited linearly exact reconstruction. All the following computations were performed using Intel Xeon 3 GHz with 2 GB RAM Memory.

10.2.3 Numerical results at $M = 0.1$

At such a Mach number, the AUSM+(P) scheme provides a very correct solution since, as one can see in the figure 10.2.3, the isolines of the Mach obtained on the finest grid remain symmetrical with respect to the sinebump.

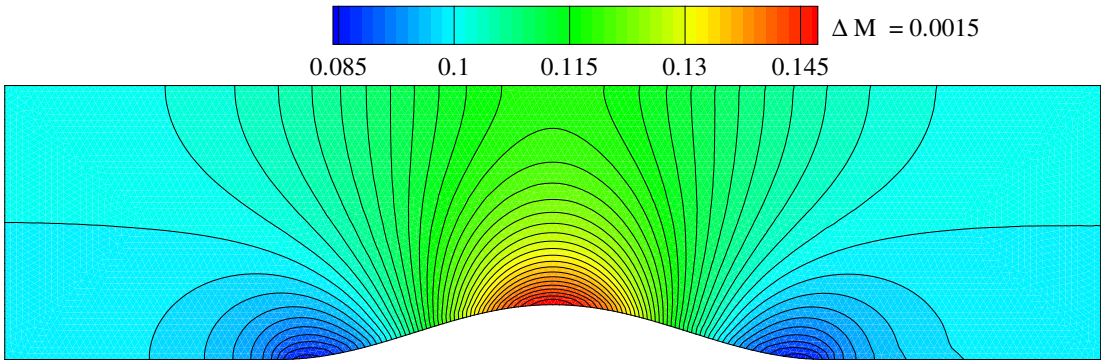


Figure 10.2.2: Stationary flow over a sine bump at $M_\infty = 0.1$. Isolines of the Mach number obtained on the finest grid.

In order to perform comparisons between the different methods described earlier, the criterion which is used to determine the convergence rate is first defined. We retain the L_2 norm of the residual :

$$Res(n) = \frac{\sqrt{\sum_{i=1}^{NBEL} (\Delta w_i^n)^2}}{\sqrt{\sum_{i=1}^{NBEL} \left(\Delta w_i^1\right)^2}}$$

where "NBEL" is the number of elements and w denotes either the density, the momentum or the energy.

In table 10.2.1, we present the results that we obtained with these different methods. Let us precise the meaning of some acronyms. "NBEL" is the number of elements, "Iterations" and "CPU" are respectively the number of non-linear iterations and the CPU time required to achieve convergence, "CPI" represents the cost per iteration while "CPPPI" denotes the cost per iteration and per element, finally, "MEO" is the fraction of CPU time consumed by the Most Expensive Operator. In the case of the Matrix-Free schemes, most of the CPU time is spent during the sub-iterative process of the relaxation procedure (Point Jacobi or Symmetric Gauss-Seidel) while in the case of Newton-Krylov algorithm, the CPU time is mainly consumed to find the solution of the involved linear system.

As expected, Newton Krylov algorithm requires few iterations to achieve the convergence. In this table, one can notice that, for the coarse grid, the NK scheme is strangely less efficient than for finer grids since it requires 120 non-linear iterations to reach the steady state while it takes only 45, 75 and 65 on the other finer grids. The reasons of such a behaviour are not well understood. At any case, the NK scheme provides a very good intrinsic efficiency, however, as its

cost per iteration is highly expensive, the global computational cost is not so small. Moreover, it is noticeable that the cost per point per iteration (CPPPI) rises as the grid becomes finer. This cost can be reduced a little by diminishing the amount of linear residuum required to achieve the iterative process. Indeed, the NK(3) scheme provides the same efficiency than the NK(10) scheme for a reduced computational cost. As a result, we did not perform the computation with the NK(10) scheme on the finest grid.

On the other hand, the Matrix-Free schemes need a large amount of iteration to achieve the convergence and, as forecasted by the Von Neumann analysis, this amount increase as the grid becomes finer. In order to maintain a good damping of the low-frequency error modes, it is necessary to increase the number of sub-iterations, especially for the Point-Jacobi algorithm (*cf.* chapter 5). However, as the costs per iteration of the Matrix-Free schemes are dramatically small, these schemes are globally more efficient than their Newton-Krylov counterparts.

In figure 10.2.3, the convergence history with respect to the number of iterations and with respect to the CPU time is presented for the finest grid. We can clearly observe the strong intrinsic efficiency of the NK scheme but, thanks to their very cheap costs per iteration, the MF schemes are more competitive in terms of CPU time.

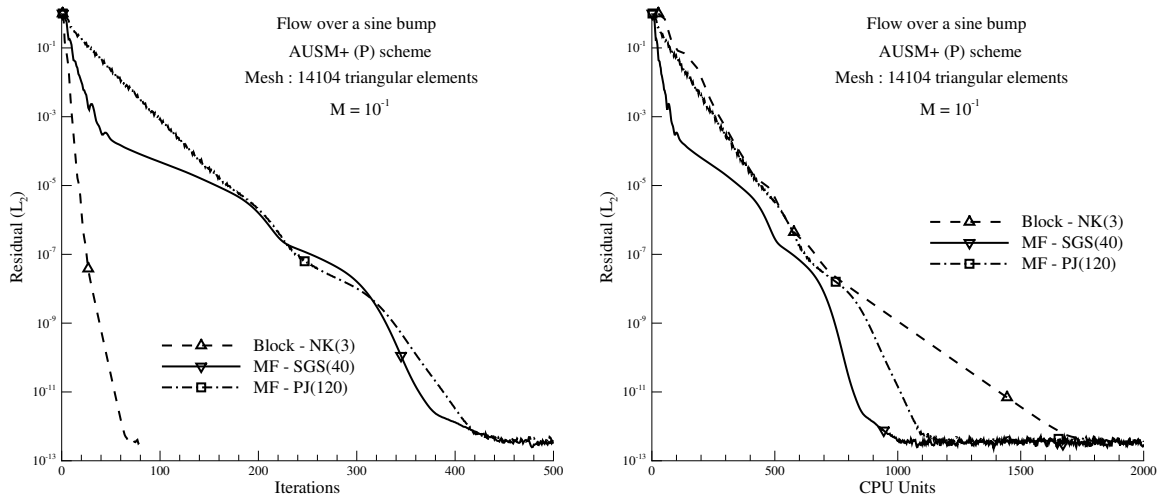


Figure 10.2.3: Stationary flow over a sine bump at $M_\infty = 0.1$. Convergence history with respect to the number of non-linear iterations (Left) and with respect to the CPU time (Right) for the finest grid.

10.2.4 Cancellation error problem

Before studying the performance properties of each method at $M = 10^{-4}$, let us deal with the problem of the cancellation errors that occurs when the Mach number is very small. As the Mach number decreases, we observe that the convergence of the preconditioned scheme does not reach the zero machine. In figure 10.2.4, we clearly notice such a behaviour. First propositions to circumvent this issue can be found in [14] but further explanations and solutions were given by Müller in [58] and by Sesterhenn *et al.* in [70]. In fact, for low Mach number problem, the average pressure is of the order of unity while its spatial variation is of the order of square of the Mach number. Then, the more the Mach number decreases, the more the

NBEL	Method	Iterations	CPU	CPI	CPPPI	MEO
878	MF-PJ(50)	200	14 s.	0.07	$8 \cdot 10^{-5}$	79 %
	MF-SGS(15)	190	**12 s.**	0.06	**7.2** $\cdot 10^{-5}$	76 %
	NK(10)	**120**	71 s.	0.6	$67 \cdot 10^{-5}$	72 %
	NK(3)	**120**	51 s.	0.43	$48 \cdot 10^{-5}$	62 %
3468	MF-PJ(80)	290	127 s.	0.44	$12.6 \cdot 10^{-5}$	90 %
	MF-SGS(20)	300	**92 s.**	0.31	**8.8** $\cdot 10^{-5}$	86 %
	NK(10)	**45**	188 s.	4.2	$120 \cdot 10^{-5}$	88 %
	NK(3)	**45**	120 s.	2.7	$77 \cdot 10^{-5}$	82 %
7898	MF-PJ(100)	380	474 s.	1.25	**15.8** $\cdot 10^{-5}$	93 %
	MF-SGS(40)	370	**472 s.**	1.28	$16 \cdot 10^{-5}$	93 %
	NK(10)	**75**	1156 s.	15	$195 \cdot 10^{-5}$	93 %
	NK(3)	**75**	682 s.	9.1	$115 \cdot 10^{-5}$	93 %
14104	MF-PJ(120)	430	1120 s.	2.6	$18.5 \cdot 10^{-5}$	94 %
	MF-SGS(40)	430	**970 s.**	2.27	**16.1** $\cdot 10^{-5}$	93 %
	NK(10)	–	– s.	–	–	– %
	NK(3)	**65**	1707 s.	26	$184 \cdot 10^{-5}$	93 %

Table 10.2.1: Stationary flow over a sine bump. $M = 0.1$. NBEL is the number of elements, Iterations is the number of non-linear iterations to achieve the convergence, CPU represents the CPU time, CPI is the cost per iteration, CPPPI is the cost per iteration and per element, MEO is the fraction of CPU time consumed by the "Most Expensive Operator".

spatial variation of the pressure becomes small with respect to its average value, and the more a numerical algorithm involving a single pressure becomes intrinsically inaccurate because of the cancellation error. For instance, in our case, the computed variation of the pressure is about 10^{-8} at $M = 10^{-4}$. Then, if the pressure converges to the zero machine, namely 10^{-15}, the spatial pressure variation is only accurate till the seven-th digit. This round-off error affects the equations and, consequently, the convergence error cannot reaches the zero machine. In order

to circumvent this issue, it is necessary to decouple the thermodynamic pressure from its spatial dynamic part so as to eliminate it from the momentum equation. We introduced such corrections in our Finite-Difference code and we succeeded in reach the machine zero for Mach numbers of about 10^{-8}. The corrections of the CAST3M code require a large amount of modifications and precautions. Besides, the flow speeds that we are interested in are usually of the order of $M = 10^{-4}$ for which the accuracy is sufficient. Thus, we did not proceed to these modifications. Nevertheless, it would be profitable to implement the corrections in the future.
In the figure 10.2.4, we can also notice that, although the minimum level of the residual increases because of the cancellation errors, the convergence rate does not vary with respect to the Mach number. This emphasizes a well-known property of the preconditioning techniques which remains valid for the Matrix-Free implicit treatment.

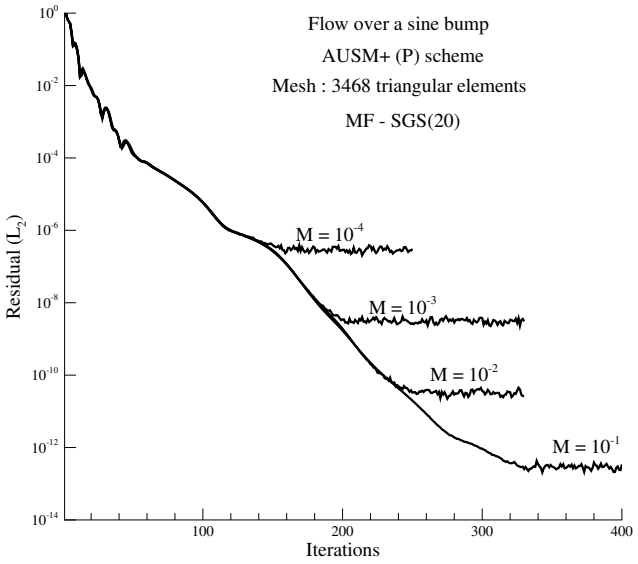


Figure 10.2.4: Stationary flow over a sine bump. Convergence history of the MF-SGS(20) scheme on the second grid (3468 elements) for several Mach numbers. Emphasis of the cancellation error problem.

10.2.5 Numerical results at $M = 10^{-4}$

In the manner of the previous paragraph, we present in figure 10.2.5 the isolines of the Mach number obtained on the very fine grid. The solution remains symmetrical for this low-Mach number case confirming the good properties of the AUSM+(P) scheme in the quasi-incompressible regime.

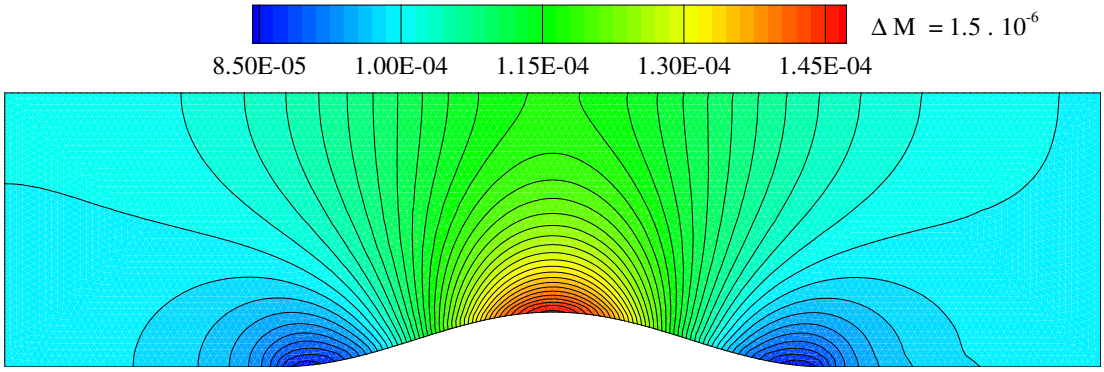


Figure 10.2.5: Stationary flow over a sine bump at $M_\infty = 10^{-4}$. Isolines of the Mach number obtained on the finest grid.

The independence of the convergence rate with respect to the Mach number, that we showed earlier, is valid for all the implicit schemes. Besides, owing to the cancellation errors, it takes less iterations for every method to achieve the convergence. For instance, in figure 10.2.4, it is noticeable that, for the MF-SGS(20) scheme on the second grid, the residual stagnates after 160 iterations at $M = 10^{-4}$ while it reach the machine accuracy after 300 iterations at $M = 10^{-1}$. These results are aslo confirm in table 10.2.2. As a result, CPU time consumption is smaller. Nevertheless, one can notice that for the NK schemes the cost per point and per iteration increases a little with respect to the case at $M = 0.1$. Indeed, we observed that the NK method became less robust when the Mach number decreased and required more linear iterations to reach the same level of linear residuum. It explains why the CPPPI is bigger than for the previous case. Moreover, for the finest grid, this lack of robustness cannot be circumvented by enlarging the size of the ILUT matrix because it then requires too much RAM memory. Thus, the question of the memory requirements is a big issue even for a classical 2D test-case.
On the other hand, the cost per point per iteration does not vary much in the case of the Matrix-Free schemes. They remain robust at small Mach number and, as noticed in the previous section, MF-PJ and MF-SGS schemes provide quite the same efficiency. Moreover, they are more competitive than their Newton-Krylov counterpart and it is noticeable that it takes less CPU time to simulate the flow over the finest grid with the Matrix-Free schemes than to compute the same flow over the third grid with the NK(3) algorithm. Furthermore, we did not meet any issue with the memory requirements.

NBEL	Method	Iterations	CPU	CPI	CPPPI	MEO
878	MF-PJ(50)	100	**7.6 s.**	0.076	**8.6** $\cdot 10^{-5}$	79 %
	MF-SGS(15)	100	7.8 s.	0.078	$8.9 \cdot 10^{-5}$	76 %
	NK(10)	**40**	44 s.	1.1	$125 \cdot 10^{-5}$	84 %
	NK(3)	**40**	32 s.	0.8	$91 \cdot 10^{-5}$	78 %
3468	MF-PJ(80)	150	69 s.	0.46	$13.3 \cdot 10^{-5}$	90 %
	MF-SGS(20)	160	**52 s.**	0.33	**9.4** $\cdot 10^{-5}$	85 %
	NK(10)	**22**	184 s.	8.4	$241 \cdot 10^{-5}$	94 %
	NK(3)	**22**	146 s.	6.6	$191 \cdot 10^{-5}$	92 %
7898	MF-PJ(100)	180	228 s.	1.27	**16** $\cdot 10^{-5}$	93 %
	MF-SGS(40)	175	**227 s.**	1.3	$16.4 \cdot 10^{-5}$	93 %
	NK(10)	**22**	1058 s.	48	$610 \cdot 10^{-5}$	98 %
	NK(3)	**22**	590 s.	27	$340 \cdot 10^{-5}$	96 %
14104	MF-PJ(120)	225	585 s.	2.6	$18.4 \cdot 10^{-5}$	93 %
	MF-SGS(40)	**220**	**514 s.**	2.34	**16.6** $\cdot 10^{-5}$	93 %
	NK(10)	–	– s.	–	–	– %
	NK(3)	NEM*	– s.	–	–	– %

Table 10.2.2: Stationary flow over a sine bump. $M = 10^{-4}$. NBEL is the number of elements, Iterations is the number of non-linear iterations to achieve the convergence, CPU represents the CPU time, CPI is the cost per iteration, CPPPI is the cost per iteration and per element, MEO is the fraction of CPU time consumed by the "Most Expensive Operator". NEM means Not Enough Memory.

10.2.6 Numerical results at $M = 0.5$

In this section, we study the performances of our Matrix-Free implicit treatment in the case of subsonic flows. We then set the Mach number at $M = 0.5$ and we compare the method with the Newton-Krylov approach. The results are given in table 10.2.3 and we can observe that, compare to the low-Mach number cases, the Matrix-Free schemes require a larger amount of iterations to reach the steady state. This is in good agreement with the Von Neumann analysis that we performed in Chapter 5. For subsonic flows, the eigenvalues take very disparate values so that the spectral radius simplification is very penalizing for the convergence rate. On the other hand, when the preconditioning matrix is applied at small Mach number regimes the condition number is very close to unity which ensures that the Matrix-Free simplification does not really deteriorate the efficiency. Moreover, we note that the NK approach also requires more iterations

NBEL	Method	Iterations	CPU	CPI	CPPPI	MEO
878	MF-PJ(50)	390	29 s.	0.074	$8.5 \cdot 10^{-5}$	78 %
	MF-SGS(15)	340	**22 s.**	0.065	**7.37** $\cdot 10^{-5}$	75 %
	NK(3)	**122**	49 s.	0.4	$46 \cdot 10^{-5}$	60 %
3468	MF-PJ(80)	410	184 s.	0.45	$12.9 \cdot 10^{-5}$	90 %
	MF-SGS(20)	420	**136 s.**	0.33	**9.4** $\cdot 10^{-5}$	85 %
	NK(3)	**65**	158 s.	2.4	$70 \cdot 10^{-5}$	80 %
7898	MF-PJ(100)	480	**604 s.**	1.26	**15.9** $\cdot 10^{-5}$	93 %
	MF-SGS(40)	480	624 s.	1.3	$16.4 \cdot 10^{-5}$	93 %
	NK(3)	**90**	714 s.	8	$100 \cdot 10^{-5}$	87 %
14104	MF-PJ(100)	650	1455 s.	2.24	**15.9** $\cdot 10^{-5}$	93 %
	MF-SGS(40)	610	**1432 s.**	2.34	$16.6 \cdot 10^{-5}$	93 %
	NK(3)	**75**	1443 s.	19	$136 \cdot 10^{-5}$	91 %

Table 10.2.3: Stationary flow over a sine bump. $M = 0.5$. NBEL is the number of elements, Iterations is the number of non-linear iterations to achieve the convergence, CPU represents the CPU time, CPI is the cost per iteration, CPPPI is the cost per iteration and per element, MEO is the fraction of CPU time consumed by the "Most Expensive Operator".

to converge. However this increase is less important than for the MF schemes. Indeed, when the ratio in terms of iterations between the MF-SGS scheme and the NK algorithm is of about 6.6 for

the finest mesh at $M = 0.1$, it reaches 8.1 at $M = 0.5$. However, these loss of intrinsic efficiency is just sufficient to make the NK approach as competitive as its Matrix-Free counterparts.
The figure 10.2.6 presents the isovalues of the Mach number for this subsonic case. The solution is still symmetric since the Mach number remains smaller than unity at the nozzle.

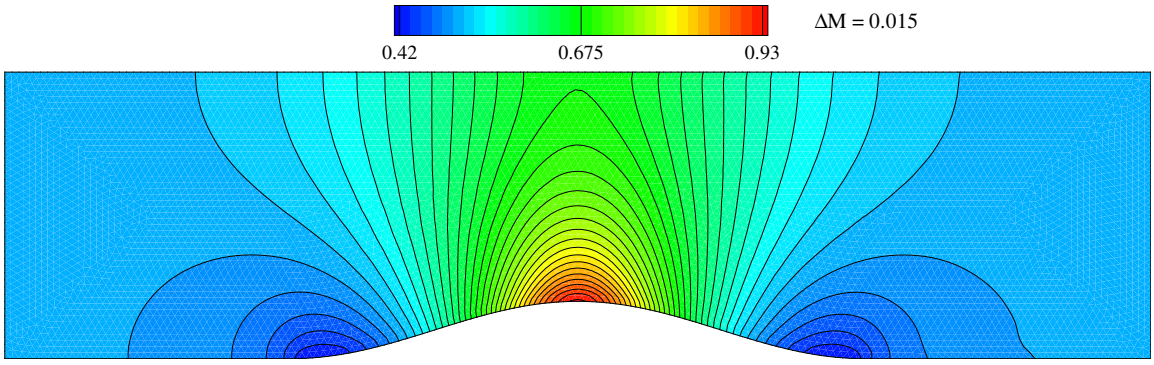


Figure 10.2.6: Stationary flow over a sine bump at $M_\infty = 0.5$. Isolines of the Mach number obtained on the finest grid.

10.2.7 Conclusions

For all the configurations that were studied, the Matrix-Free implicit treatment designed for the unstructured grids was shown to be as efficient as its Cartesian version. This new implicit scheme, coupled with either SGS or PJ relaxation procedure, enables to reduce dramatically the cost of the computations with respect to the former Newton-Krylov implicit algorithm especially for the low-Mach number flow. Moreover, even if the case considered was only 2D, we had to cope with the issue of the memory requirement for the NK approach. On the other hand, the simplicity of the Matrix-Free method allows to perform fast simulations with regular computers. As forecasted by the Von Neumann analysis and the Cartesian test-cases, the MF-SGS scheme ensures a very good convergence rate even when the mesh becomes very fine thanks to its good damping properties over the low-frequency region, while the MF-PJ scheme requires more inner-iterations so as to eliminate efficiently all the error modes. However, the latter remains a competitive alternate approach for computing all speed flows over unstructured grids. Convergence performances will be now studied for 3D configurations for which the question of the memory requirements will be a crucial issue. The next section will aim at outlining whether the Matrix-Free schemes enable to compute flow over grids involving a large amount of points or not.

10.3 3D Inviscid Sinebump Channel

This test-case is a three-dimensional simulation of a two-dimensional flow. It is used to emphasize the efficiency and the robustness of the new Matrix-Free implicit treatment. The initial and boundary conditions remain the same with respect to the previous test-case. The geometry is now a parallelepiped $(x, y, z) \in [0; 4] \times [0; 0.5] \times [0; 1]$ with a straight upper wall and a curve lower wall :

$$\begin{cases} x < 1 & z = 0 \\ 1 \geq x \geq 3 & z = 0.1\ (1 - \cos[(x-1)\pi]) \\ x > 3 & z = 0 \end{cases}$$

The lateral walls of the computational domain correspond to the surface of the 2D domain. Hence, four different grids using six-face prisms are available as for the 2D test-case : a coarse one with 4390 elements and 12103 faces (*cf.* figure 10.3), a medium one with $34,780$ elements and $91,428$ faces, a fine one with $118,470$ elements and $306,323$ faces, and a very fine one with $282,080$ elements and $723,304$ faces.

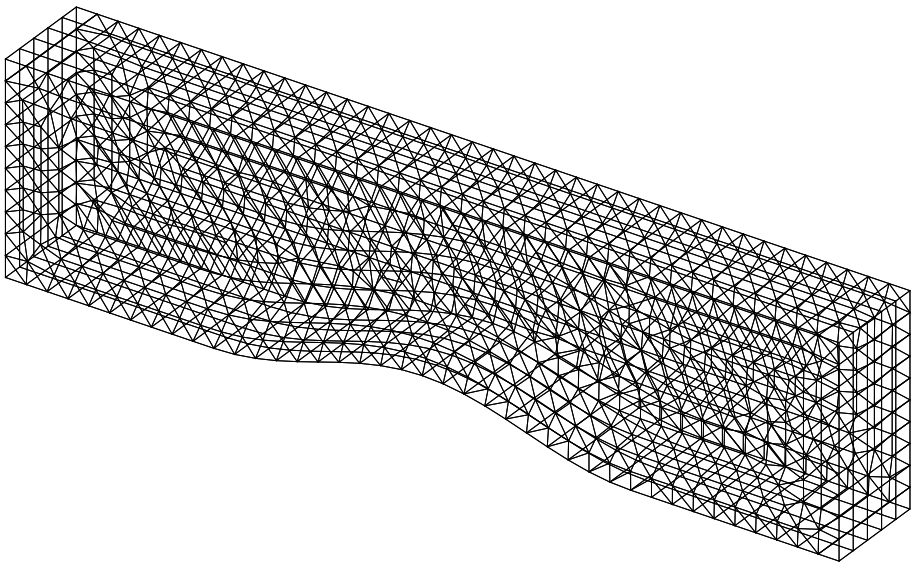


Figure 10.3.1: 3D inviscid sinebump channel. Coarse unstructured grid with $4,390$ elements (prisms) and $12,103$ faces.

The solution of this test-case, as for its 2D counterpart, is symmetrical and isopotential. In figure 10.3, the isoligns of the Mach number (top) and the isobaric curves (bottom) are presented. They were computed with the Matrix-Free scheme over the finest grid at $M_\infty = 0.1$. This numerical solution computed with the AUSM+(P) approach is in good agreement with the theoretical solution.

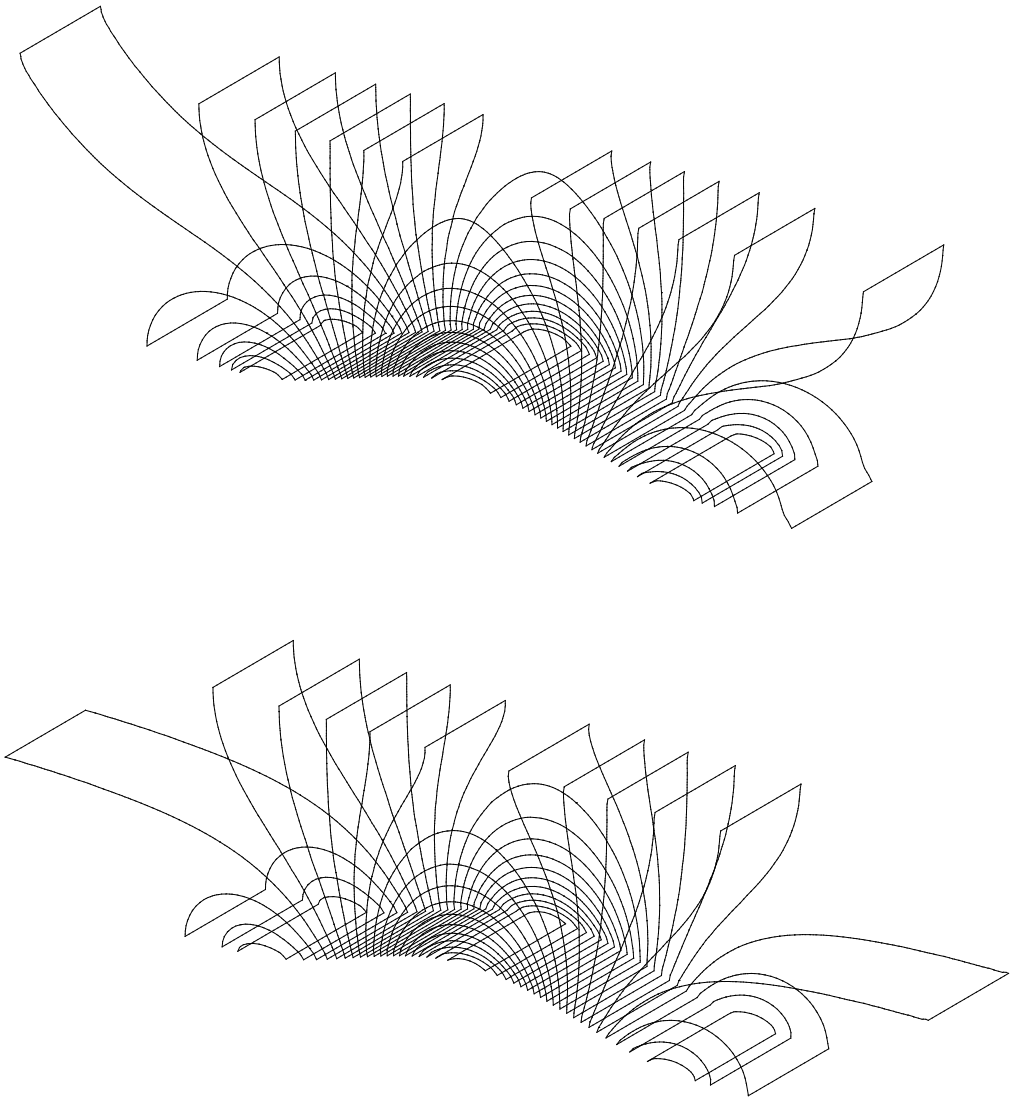


Figure 10.3.2: Stationary flow in a 3D sinebump channel at $M_\infty = 0.1$ and over the finest grid (282080 elements). Top : Isoligns of the Mach number. Bottom : Isobaric curves.

The results are summarized in the table 10.3.1. As expected the Newton-Krylov is highly limited by its prohibited memory requirements. Indeed, from the second grid with only 34,780 elements, the memory storage is too big for the capacities of regular computers. Moreover, for the coarsest mesh, the method suffers from its great cost per iteration which makes it totally inefficient compare to the Matrix-Free schemes.

NBEL	Method	Iterations	CPU	CPI	CPPPI	MEO
4,390	MF-PJ(50)	210	**149 s.**	0.71	**1.6** $\cdot 10^{-4}$	83 %
	MF-SGS(20)	220	153 s.	0.71	**1.6** $\cdot 10^{-4}$	83 %
	NK(3)	**120**	17500 s.	146	$332 \cdot 10^{-4}$	95 %
34,780	MF-PJ(80)	**310**	**2570 s.**	8.3	**2.4** $\cdot 10^{-4}$	85 %
	MF-SGS(40)	340	3180 s.	9.3	$2.7 \cdot 10^{-4}$	88 %
	NK(3)	NEM*	– s.	–	–	– %
118,470	MF-PJ(100)	**410**	**15000 s.**	37	**3.1** $\cdot 10^{-4}$	85 %
	MF-SGS(50)	460	18800 s.	41	$3.5 \cdot 10^{-4}$	89 %
282,080	MF-PJ(120)	**580**	**62400 s.**	107	$3.8 \cdot 10^{-4}$	86 %
	MF-SGS(50)	630	65000 s.	103	**3.7** $\cdot 10^{-4}$	89 %

Table 10.3.1: Stationary flow over a 3D-sine bump. $M = 0.1$. NBEL is the number of elements, Iterations is the number of non-linear iterations to achieve the convergence, CPU represents the CPU time, CPI is the cost per iteration, CPPPI is the cost per iteration and per element, MEO is the fraction of CPU time consumed by the "Most Expensive Operator". NEM means Not Enough Memory.

On the other hand, the Matrix-Free technique enables to deal with grids involving a very large number of elements with only 2 GB of RAM memory. In addition, the method provides a truly competitive efficiency since computing the solution over the third grid takes as much CPU time as for simulating the same flow on the coarsest grid with the NK approach. The objective of reducing the memory requirements without deteriorating the efficiency is therefore reached.

As far as the Matrix-Free schemes are concerned, the hierarchy obtained for the 3D case is not the same than the one derived from the 2D case. Indeed, it is noticeable that the MF-PJ scheme is the more efficient. In fact, the convergence rate of the MF-SGS approach deteriorates with respect to the 2D case while it remains the same for the MF-PJ technique. This loss of intrinsic efficiency is accentuated by an increase of the CPPPI. Such a behaviour can be explained by the fact that the intrinsic efficiency of the MF-SGS relies on the ordering of the mesh cells (*cf.* [72]). As no re-ordering technique devoted to this technique is available in the CAST3M code,

it is necessary to build the grid carefully in order to improve the efficiency. However, it does not circumvent the problem of the increase of the CPPPI. As the SGS algorithm requires the knowledge of the connectivity of the mesh cells and as the number of cells rises a lot for the 3D cases, the cost of the method becomes more expensive. On the other hand, the Point Jacobi procedure is not dependent on the ordering of the mesh cells and until the third mesh, its cost per iteration remains quasi constant. This method seems therefore well-suited to perform fast 3D computations. However, it is noticeable that we should increase strongly the number of inner-sweeps for the finest grids so as to damp correctly the low-frequency error modes. The SGS algorithm becomes then competitive to carry out 3D calculations over grids involving very small cells as shown in figure 10.3.

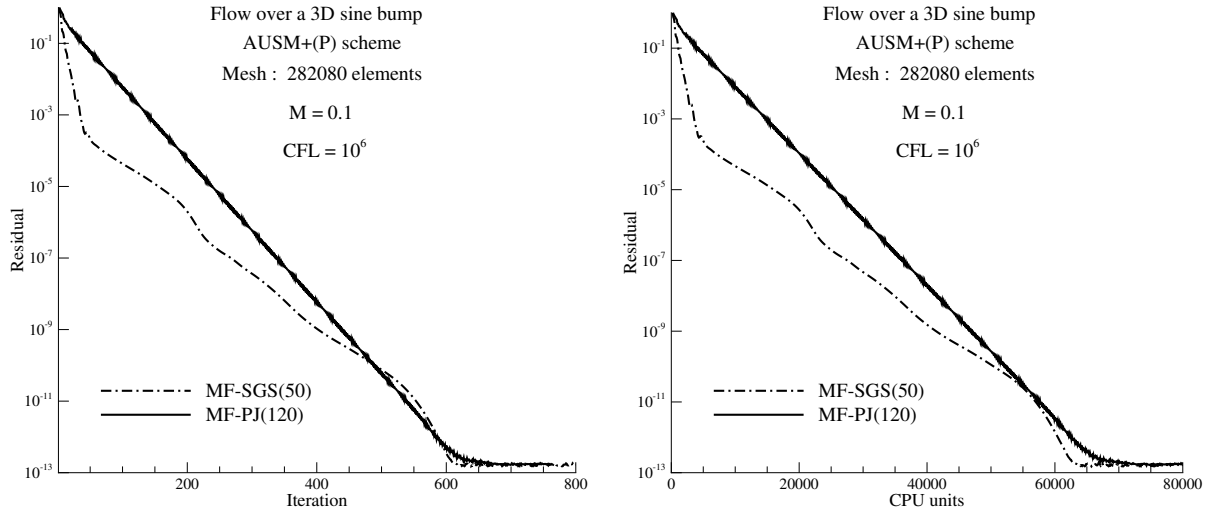


Figure 10.3.3: Stationary flow in a 3D sinebump channel at $M_\infty = 0.1$ over the finest grid (282080 elements). Convergence history with respect to the number of iterations (left) and with respect to the CPU time (right).

10.4 Conclusions

This chapter was devoted to the comparisons between the implicit treatments available in the CAST3M code over 2D and 3D inviscid test-cases. We aimed at confirming the very good properties of the Matrix-Free framework in the case of unstructured grids. This method was compared to the Newton-Krylov approach involving a GMRES algorithm preconditioned using an ILUT technique.

The computation of an inviscid flow in a 2D sinebump channel at subsonic and low-Mach number regime demonstrated that the newly implemented Matrix-Free approach was globally as efficient as its Newton-Krylov block counterpart. Moreover, even for this simple 2D test-case, we had to cope with the issue of the memory requirement for the NK approach. Indeed, a certain lack of robustness obliged us to increase the size of the preconditioner ILUT which could not then be treated by regular computers. On the other hand, the simplicity of the Matrix-Free method

allowed to perform fast simulations for all the configurations. Both MF-PJ and MF-SGS schemes provides good convergence rates; however, it is necessary to increase the number of inner-sweeps, especially for the PJ procedure, in order to eliminate efficiently all the error modes when the mesh becomes finer. In spite of this large amount of inner-iterations, both methods remain very competitive to compute inviscid stationary flows.
The 3D test-case aimed at checking the ability of the Matrix-Free implicit treatment to deal with grids involving a large amount of points. We then built a 3D version of the previous 2D test-case. As expected, the Newton-Krylov scheme was quickly limited because of its huge memory requirements. Indeed, the memory resources of regular computers (2GB RAM memory) are not sufficient to deal with more than $10,000$ elements in 3D. On the other hand, using Matrix-Free schemes allowed to compute flow over grids involving more than $592,350$ elements on the same machines. In a 3D context, the MF-PJ scheme is surprisingly the most efficient. In fact, the efficiency of its MF-SGS counterpart relies on the ordering of the cells and, moreover, it requires the knowledge of the connectivity of the cells which contributes to increase the cost per iteration. The MF-PJ method seems to be a very competitive approach to perform fast 3D computations. This technique is also very attractive since, as it is said in [72], it can be easily implemented in a parallel code.

Chapter 11

Modelling of a Tee Junction at Low-Mach Number Regime

11.1 Introduction

The mixing of an hot and cold fluid in a tee junction (simplified scheme in figure 11.1.1) has been the object of several investigations in Nuclear Energy Industry (see for instance [33, 32]). Indeed in the region in which hot and cold flow streams join, random fluctuations of the coolant

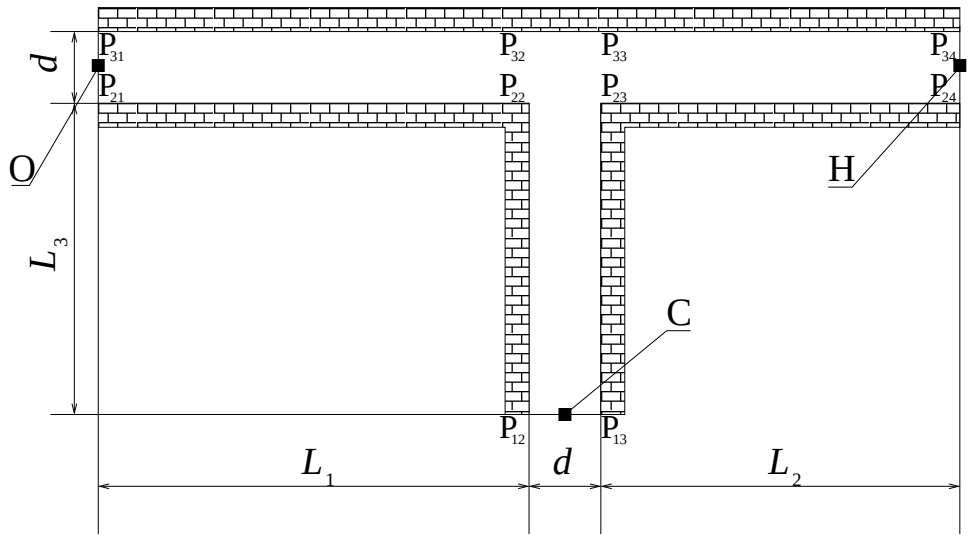


Figure 11.1.1: Tee junction. H represents the inlet of the hot fluid, C of the cold one and O is the outlet.

temperature can occur. Such fluctuations can cause cyclical thermal stresses (already known as thermal striping) and subsequent fatigue cracking of the piping. Accidents linked to thermal striping already occurred at some nuclear power plants (Oskarshamn/Ringhals1 /Barsabeck2 in Sweden, Tsuruga in Japan, Civaux in France), which increased the interest on this phenomenon. As mentioned in [13], such accidents are very complex to analyze. Indeed several fields are involved: thermal-hydraulics, heat transmission, mechanics and material science. Each field has

its limits, which makes this phenomenon difficult to understand.
The understanding of thermal stripping is beyond the purpose of this study. In fact, this chapter displays a comparison between the asymptotic solver available in CAST3M [61] and the newly implemented low-Mach matrix-free treatment for the case of laminar, low-Mach number, steady and unsteady flows in a 2D Tee junction. The study aims at confirming the interest of the new implicit algorithm for efficient computation of low-Mach number (single-species) flows.

11.2 The steady problem

Let us present the stationary problem we solve. We suppose to deal with a calorically perfect gas (for instance air). As shown in figure 11.1.1, such gas enters the inlet H with temperature T_H and momentum $\tilde{m}_H$ and the inlet C with temperature T_C and momentum $\tilde{m}_C$. The inlet temperatures are constant in space and time. The inlet momenta are parabolic, symmetric and parallel with respect to the tube axis (zero at the surfaces):

$$\tilde{m}_H(s) = \frac{6\bar{\tilde{m}}_H}{d^2}\left(\frac{d^2}{4} - s^2\right) \quad , \quad \tilde{m}_C(s) = \frac{6\bar{\tilde{m}}_C}{d^2}\left(\frac{d^2}{4} - s^2\right) \quad ,$$

s being the distance with respect to the tube axis, $\bar{\tilde{m}}_H$ and $\bar{\tilde{m}}_C$ the average momentums. At the outlet O, the pressure is constant (in space and time) and equal to P_0 (plus the hydrostatic component). Walls are impermeable and adiabatic everywhere. The velocity is zero at the wall. For the sake of simplicity, we consider constant dynamic viscosity and thermal diffusivity.

11.3 Governing equations

We consider the Navier-Stokes equations for compressible, calorically perfect gas, namely :

$$\begin{cases} \dfrac{\partial \tilde{\rho}}{\partial \tilde{t}} + \vec{\tilde{\nabla}} \cdot (\tilde{\rho}\vec{\tilde{u}}) = 0 \\ \dfrac{\partial \tilde{\rho}\vec{\tilde{u}}}{\partial \tilde{t}} + \vec{\tilde{\nabla}} \cdot (\tilde{\rho}\vec{\tilde{u}} \otimes \vec{\tilde{u}} + \tilde{p}\underline{I}) = \tilde{\rho}\vec{\tilde{g}} + \vec{\tilde{\nabla}} \cdot \underline{\tilde{\tau}} \\ \dfrac{\partial \tilde{\rho}\tilde{E}}{\partial \tilde{t}} + \vec{\tilde{\nabla}} \cdot (\tilde{\rho}\vec{\tilde{u}}\tilde{H}) = \tilde{\rho}\vec{\tilde{g}} \cdot \vec{\tilde{u}} + \vec{\tilde{\nabla}} \cdot (\underline{\tilde{\tau}} \cdot \vec{\tilde{u}}) + \vec{\tilde{\nabla}} \cdot (\tilde{\lambda}\vec{\tilde{\nabla}}\tilde{T}) \\ \tilde{p} = \tilde{\rho}R\tilde{T} = (\gamma - 1)\tilde{\rho}\left(\tilde{E} - \dfrac{1}{2}\vec{\tilde{u}} \cdot \vec{\tilde{u}}\right) \end{cases} \tag{11.3.1}$$

where

$$\underline{\tilde{\tau}} = \tilde{\mu}\left(\vec{\tilde{\nabla}} \otimes \vec{\tilde{u}} + \left(\vec{\tilde{\nabla}} \otimes \vec{\tilde{u}}\right)^T - \frac{2}{3}\left(\vec{\tilde{\nabla}} \cdot \vec{\tilde{u}}\right)\underline{I}\right)$$

Let us carry out the non-dimensionalization of the equations (11.3.1). First of all, we define the following coefficients :

$$\Delta\tilde{T} = \tilde{T}_H - \tilde{T}_C \quad , \quad \bar{\tilde{T}} = \frac{\tilde{T}_H + \tilde{T}_C}{2} \quad , \quad \epsilon = \frac{\Delta\tilde{T}}{2\bar{\tilde{T}}} \quad , \quad \bar{\tilde{m}}_t = \bar{\tilde{m}}_H + \bar{\tilde{m}}_C \quad , \quad \alpha_H = \frac{\bar{\tilde{m}}_H}{\bar{\tilde{m}}_t} \quad . \tag{11.3.2}$$

Then, the reference values are chosen as follows :

$$\tilde{l}_{ref} = d \quad , \quad \tilde{P}_{ref} = \tilde{P}_0 \quad , \quad \tilde{\rho}_{ref} = \frac{\tilde{P}_0}{R\bar{\tilde{T}}} \quad , \quad \tilde{u}_{ref} = \frac{\bar{\tilde{m}}_t}{\tilde{\rho}_{ref}} \quad , \quad \tilde{t}_{ref} = \frac{d}{\tilde{u}_{ref}} \quad ,$$
$$\tilde{\lambda}_{ref} = \tilde{\lambda}(\tilde{\rho}_{ref}, \tilde{P}_{ref}) \quad , \quad \tilde{\mu}_{ref} = \tilde{\mu}(\tilde{\rho}_{ref}, \tilde{P}_{ref}) \quad . \tag{11.3.3}$$

In the system (11.3.1), dimensional values are replaced by :

$$\vec{r} = \frac{\tilde{\vec{r}}}{\tilde{l}_{ref}} \quad , \quad t = \frac{\tilde{u}_{ref}}{\tilde{l}_{ref}}\tilde{t} \quad , \quad u = \frac{\tilde{u}}{\tilde{u}_{ref}} \quad , \quad p = \frac{\tilde{p}}{\tilde{P}_{ref}} \quad , \quad \rho = \frac{\tilde{\rho}}{\tilde{\rho}_{ref}} \quad , \quad \rho e_t = \frac{\tilde{\rho}\tilde{e}_t}{\tilde{P}_{ref}} \quad ,$$
$$T = \frac{\tilde{T} - \bar{\tilde{T}}}{\Delta\tilde{T}} \quad , \quad \lambda = \frac{\tilde{\lambda}}{\tilde{\lambda}_{ref}} \quad , \quad \mu = \frac{\tilde{\mu}}{\tilde{\mu}_{ref}} \quad , \quad \tau = \frac{\tilde{l}_{ref}}{\tilde{\mu}_{ref}\tilde{u}_{ref}}\tilde{\tau} \quad . \tag{11.3.4}$$

After straightforward calculations, the following system of equations is found :

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{u}) = 0 \\ \dfrac{\partial \rho \vec{u}}{\partial t} + \vec{\nabla} \cdot (\rho \vec{u} \otimes \vec{u}) + \dfrac{1}{\gamma M_\infty^2} \vec{\nabla} \cdot (p \underline{I}) = \dfrac{1}{Fr}\ \rho \vec{e_g} + \dfrac{1}{Re}\ (\vec{\nabla} \cdot \underline{\tau}) \\ \dfrac{\partial \rho E}{\partial t} + \vec{\nabla} \cdot (\rho \vec{u} H) = \dfrac{\gamma M_\infty^2}{Fr}\ \rho \vec{e_g} \cdot \vec{u} + \dfrac{\gamma M_\infty^2}{Re}\ \vec{\nabla} \cdot (\underline{\tau} \cdot \vec{u}) + 2\epsilon \dfrac{\gamma}{\gamma - 1}\ \dfrac{1}{Pr\ Re}\ \vec{\nabla} \cdot (\lambda \vec{\nabla} T) \\ p = \rho(1 + 2\epsilon T) = (\gamma - 1)\rho \left(E - \dfrac{\gamma M_\infty^2}{2} \vec{u} \cdot \vec{u} \right) \end{cases} \tag{11.3.5}$$

where

$$M_\infty^2 = \frac{\tilde{\rho}_{ref}\tilde{u}_{ref}^2}{\gamma \tilde{P}_{ref}} = \frac{\bar{\tilde{m}}_t^2}{\gamma \tilde{P}_{ref}^2} R\bar{\tilde{T}} \quad , \quad Pr = \frac{\tilde{\mu}_{ref}}{\tilde{\lambda}_{ref}} \frac{\gamma}{\gamma - 1} R \quad , \quad Re = \frac{\bar{\tilde{m}}_t \tilde{l}_{ref}}{\tilde{\mu}_{ref}} \quad , \quad Fr = \left(\frac{\bar{\tilde{m}}_t R\bar{\tilde{T}}}{\tilde{P}_{ref}} \right)^2 \frac{1}{g\tilde{l}_{ref}} \quad . \tag{11.3.6}$$

Let us introduce the boundary conditions under dimensionless form. As the system is opened, $P(t)$ is given by the boundary conditions in the outlet O, *i.e.* $P(t) = \tilde{P}_O(t)/\tilde{P}_0$. The inlet temperatures are :

$$T_H = \frac{\tilde{T}_H - \bar{\tilde{T}}}{\Delta\tilde{T}} = \frac{\tilde{T}_H - \tilde{T}_C}{2(\tilde{T}_H - \tilde{T}_C)} = 0.5 \ , \qquad T_C = \frac{\tilde{T}_C - \bar{\tilde{T}}}{\Delta\tilde{T}} = \frac{\tilde{T}_C - \tilde{T}_H}{2(\tilde{T}_H - \tilde{T}_C)} = -0.5 \ .$$

The inlet average dimensionless momentum are given by :

$$\bar{m}_H = \bar{\rho}_H \cdot \bar{u}_H = \alpha_H \cdot \bar{\tilde{m}}_t \ , \qquad \bar{m}_C = \bar{\rho}_C \cdot \bar{u}_C = (1 - \alpha_H) \cdot \bar{\tilde{m}}_t \ .$$

The non-dimensional solution of this problem depends on the following non-dimensional parameters: L_1/d, L_2/d, and L_3/d (which are non sensibly influent in the mixing region if they are big enough), the specific heat ratio γ, the Reynolds number Re, the Prandtl number Pr, the Froude number Fr, the Mach number, the ratios α_H and ϵ. It follows that, if we take as non-dimensional parameters

L_1/d	L_2/d	L_3/d	γ	ϵ	α_H	Pr	Re	M	Fr
12	9	7	1.4	0.2	0.8	0.7	100	1/300	1/9

and we fix the numerical values

$$d = 1 \text{ m} , \quad \bar{\rho} = \frac{P_0}{R\bar{T}} = 1 \text{ kg m}^{-3} , \quad \bar{m}_t = 1 \text{ kg m}^{-2} \text{ s}^{-1} , \quad R = 288 \text{ J kg}^{-1} \text{ K}^{-1} ,$$

we can compute all the dimensional quantities involved in the problem (SI units):

L_1	L_2	L_3	$\bar{u}$	P_0	$\bar{T}$	T_H	T_C	$\bar{m}_H$	$\bar{m}_C$	μ_0	λ_0
12	9	7	1	64290	223.2	267.8	178.6	0.8	0.2	0.01	14.4

Conservation properties

The stationary solution of this problem must satisfy the divergence condition :

$$\vec{\tilde{\nabla}} \cdot \left(\tilde{\rho} \vec{\tilde{v}} \right) = 0 . \tag{11.3.7}$$

If we neglect the contribution of the heat diffusion at the inlet and at the outlet (approximation which is correct if the sections are far enough from the mixing region), it must be :

$$\int_H \mathrm{d}\tilde{S} \left(\tilde{\rho} \tilde{H} \vec{\tilde{v}} \cdot \vec{\tilde{n}} \right) + \int_C \mathrm{d}\tilde{S} \left(\tilde{\rho} \tilde{H} \vec{\tilde{v}} \cdot \vec{\tilde{n}} \right) = - \int_O \mathrm{d}\tilde{S} \left(\tilde{\rho} \tilde{H} \vec{\tilde{v}} \cdot \vec{\tilde{n}} \right) , \tag{11.3.8}$$

i.e., by neglecting the terms $O(M^2)$:

$$\int_H \mathrm{d}\tilde{S} \left(\tilde{\rho} \tilde{h} \vec{\tilde{v}} \cdot \vec{\tilde{n}} \right) + \int_C \mathrm{d}\tilde{S} \left(\tilde{\rho} \tilde{h} \vec{\tilde{v}} \cdot \vec{\tilde{n}} \right) = - \int_O \mathrm{d}\tilde{S} \left(\tilde{\rho} \tilde{h} \vec{\tilde{v}} \cdot \vec{\tilde{n}} \right) .$$

Since we are dealing with a calorically perfect gas and the so-called thermodynamic pressure is constant, it follows that :

$$\int_H \mathrm{d}\tilde{S} \left(\vec{\tilde{v}} \cdot \vec{\tilde{n}} \right) + \int_C \mathrm{d}\tilde{S} \left(\vec{\tilde{v}} \cdot \vec{\tilde{n}} \right) = - \int_O \mathrm{d}\tilde{S} \left(\vec{\tilde{v}} \cdot \vec{\tilde{n}} \right) ,$$

i.e.

$$\int \mathrm{d}\tilde{V} \left(\vec{\tilde{\nabla}} \cdot \vec{\tilde{u}} \right) = 0 . \tag{11.3.9}$$

From equation (11.3.8), we can also deduce that :

$$\tilde{m}_H \tilde{T}_H \tilde{S}_H + \tilde{m}_C \tilde{T}_C \tilde{S}_C = - \int_O \mathrm{d}\tilde{S} \left(\tilde{\rho} \tilde{H} \vec{\tilde{v}} \cdot \vec{\tilde{n}} \right)$$

If the outlet section O is far enough from the tee junction, the outlet T is constant, namely :

$$\tilde{T}_O = \frac{\tilde{m}_H \tilde{S}_H}{\tilde{m}_O \tilde{S}_O} \tilde{T}_H + \frac{\tilde{m}_C \tilde{S}_C}{\tilde{m}_O \tilde{S}_O} \tilde{T}_C = \alpha_H \tilde{T}_H + (1 - \alpha_H) \tilde{T}_C = 250 \text{ K} . \tag{11.3.10}$$

11.4 Numerical approaches

11.4.1 Fully compressible solver

The conservative compressible equations (11.3.1) are solved using the compressible density-based solver of CAST3M which employs an unstructured Finite Volume approach. As described earlier, the inviscid fluxes are computed using classical upwind schemes, extended to higher order using the primitive variable reconstruction of Barth and Jespersen, described in [2] and belonging to the MUSCL family; moreover the numerical dissipation of the inviscid schemes is corrected taking into account Low Mach number preconditioning. Viscous fluxes are approximated using a linearly-exact extension of the diamond method of Noh [60]. Concerning the time discretization, the Euler implicit scheme is used. The non-linear problem arising from the time discretization is solved via the Matrix-Free implicit method that we introduced in this thesis. Besides, we couple the MF scheme with a Symmetric Gauss-Seidel relaxation procedure.

11.4.2 Low-Mach asymptotic compressible solver

In order to provide comparisons, the Tee junction problem is also computed using a pressure-based solver of the CAST3M code. In such solver the compressible equations arising from one-time scale and one-space scale low-Mach number asymptotics are solved, namely, one has to compute the (constant in space) thermodynamic pressure $\tilde{P}(\tilde{t})$, the speed $\vec{\tilde{u}}(\vec{\tilde{r}},\tilde{t})$, the temperature $\tilde{T}(\vec{\tilde{r}},\tilde{t})$, and the dynamic pressure $\tilde{p}'((\vec{\tilde{r}},\tilde{t}))$, solution of the initial value problem :

$$\begin{cases} \dfrac{1}{\gamma\tilde{P}}\dfrac{\mathrm{d}\tilde{P}}{\mathrm{d}\tilde{t}} + \left(\vec{\tilde{\nabla}}\cdot\vec{\tilde{u}}\right) = \dfrac{\gamma-1}{\gamma\tilde{P}}\vec{\tilde{\nabla}}\cdot\left(\tilde{\lambda}\vec{\tilde{\nabla}}\tilde{T}\right) \\ \dfrac{\partial\vec{\tilde{u}}}{\partial\tilde{t}} + \left(\vec{\tilde{u}}\cdot\vec{\tilde{\nabla}}\right)\vec{\tilde{u}} = \dfrac{R\tilde{T}}{\tilde{P}}\left(-\vec{\tilde{\nabla}}\tilde{p}' + \vec{\tilde{\nabla}}\cdot\underline{\tilde{\tau}}\right) + \vec{g} \\ \dfrac{\partial\tilde{T}}{\partial\tilde{t}} + \left(\vec{\tilde{u}}\cdot\vec{\tilde{\nabla}}\right)\tilde{T} = \dfrac{\gamma-1}{\gamma}\dfrac{\tilde{T}}{\tilde{P}}\left(\dfrac{\mathrm{d}\tilde{P}}{\mathrm{d}\tilde{t}} + \vec{\tilde{\nabla}}\cdot\left(\tilde{\lambda}\vec{\tilde{\nabla}}\tilde{T}\right)\right) \end{cases} \tag{11.4.1}$$

The space discretization is performed using a Galerkin Finite Element approach. Velocity and temperature shape functions are quadratic (triangular elements with 7 degree of freedoms and quadrangular ones with 9) while pressure shape functions are linear (triangular elements with 3 degree of freedoms and quadrangular ones with 4). Concerning the time discretization, the BDF1 implicit scheme is used. The non-linear problem arising from the time discretization is solved via a fixed-point scheme :

$$\begin{cases} \dfrac{1}{\gamma\tilde{P}^n}\left(\dfrac{\mathrm{d}\tilde{P}}{\mathrm{d}\tilde{t}}\right)^n + \left(\vec{\tilde{\nabla}}\cdot\vec{\tilde{u}}^{\,n+1}\right) = \dfrac{\gamma-1}{\gamma\tilde{P}^n}\vec{\tilde{\nabla}}\cdot\left(\tilde{\lambda}\vec{\tilde{\nabla}}\tilde{T}^n\right) \\ \left(\dfrac{\partial\vec{\tilde{u}}}{\partial\tilde{t}}\right)^n + \left(\vec{\tilde{u}}^{\,n}\cdot\vec{\tilde{\nabla}}\right)\vec{\tilde{u}}^{\,n+1} = \dfrac{R\tilde{T}^n}{\tilde{P}^n}\left(-\vec{\tilde{\nabla}}\tilde{p}'^{\,n+1} + \vec{\tilde{\nabla}}\cdot\underline{\tilde{\tau}}^{\,n+1}\right) + \vec{g} \\ \left(\dfrac{\partial\tilde{T}}{\partial\tilde{t}}\right)^n + \left(\vec{\tilde{u}}^n\cdot\vec{\tilde{\nabla}}\right)\tilde{T}^{\,n+1} = \dfrac{\gamma-1}{\gamma}\dfrac{\tilde{T}^n}{\tilde{P}^n}\left(\left(\dfrac{\mathrm{d}\tilde{P}}{\mathrm{d}\tilde{t}}\right)^n + \vec{\tilde{\nabla}}\cdot\left(\tilde{\lambda}\vec{\tilde{\nabla}}\tilde{T}^{\,n+1}\right)\right) \end{cases} \tag{11.4.2}$$

As shown in [61], this algorithm has also been validated in computing the solution of a square cavity with large temperature differences.

Dimensionless boundary conditions

Unlike the fully-compressible case for which we impose the dimensionless average momentum at each inlet, here, we impose the dimensionless average velocities as follows :

$$\bar{u}_H = \frac{\tilde{\bar{u}}_H}{\tilde{u}_{ref}} = \frac{\alpha_H \tilde{\bar{m}}_t}{\tilde{\rho}_H} \frac{\tilde{\rho}_{ref}}{\tilde{\bar{m}}_t} = \alpha_H (1 + 2\epsilon T_H) \frac{1}{P} = \alpha_H (1 + \epsilon) \frac{1}{P} \ ,$$

$$\bar{u}_C = (1 - \alpha_H)(1 - \epsilon) \frac{1}{P} \ .$$

Besides, at the outlet, we impose that the velocity presents a profile which is parabolic and symmetric with respect to the tube axis. The constant coefficient involved in such parabola is determined using the divergence equation (first equation of (11.4.1)).

11.5 Numerical solutions

11.5.1 Convergence analysis

First of all let us perform a convergence analysis with respect to the mesh dimension. In the case of the pressure-based solver, the coarsest has 3135 elements (15 elements per tube section (*cf.* figure 11.5.1)) while the medium mesh has 5580 elements (20 elements per tube section) and the finest mesh has 8725 elements (25 elements per tube section). In the case of the density-

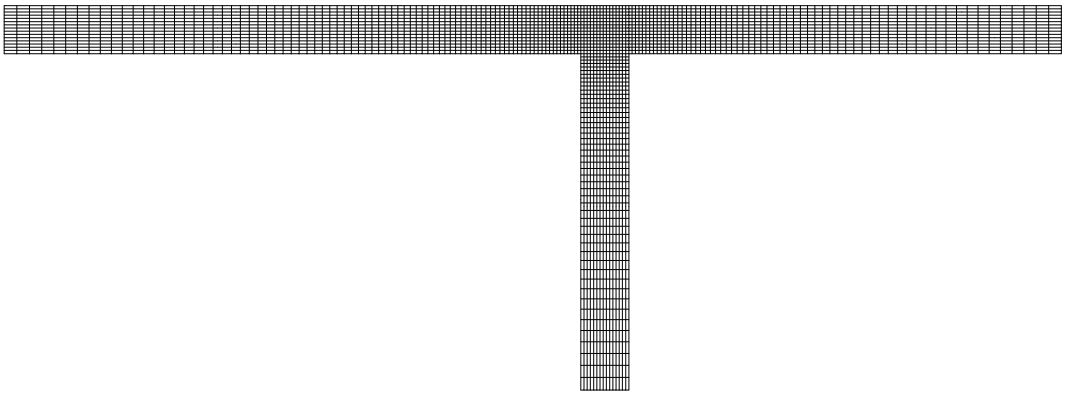


Figure 11.5.1: Tee junction. The coarse mesh (3135 elements).

based solver, we take meshes with more elements (note that neither the degrees of freedom nor the accuracy of the approach are the same as in the pressure-based solvers). The coarsest grid has 12630 elements (30 elements per tube section). The medium mesh has 22400 elements (40 elements per tube section) while the finest mesh has 35000 elements (50 elements er tube section).
In figure 11.5.2 we represent the streamlines and the temperature isolines. As one can see, the hot and the cold fluid do not mix, *i.e.* the heat transmission arrives by thermal diffusion. In figures 11.5.3, we represent the solution on the sections $P_{23}P_{33}$, $P_{22}P_{23}$ and $P_{22}P_{32}$ that we obtained using the asymptotic pressure-based compressible solver. On the other hand, in figures 11.5.4, we represent the solution on the sections $P_{23}P_{33}$, $P_{22}P_{23}$ and $P_{22}P_{32}$ that we

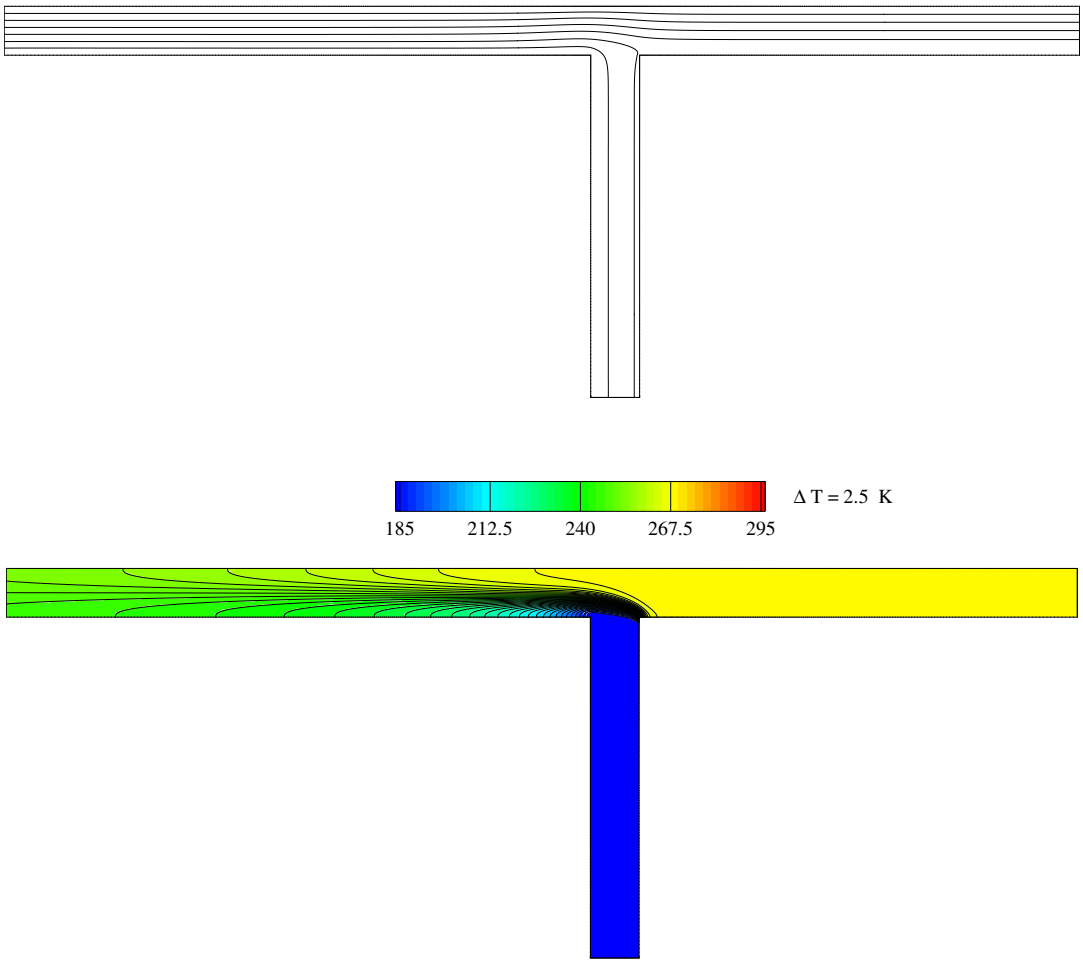


Figure 11.5.2: Tee junction. Stationary solution. Streamlines (top) and temperature isolines (bottom) given by the density-based compressible solver.

computed using the density-based compressible solver. As one can see, all the presented results are quite mesh-independent.

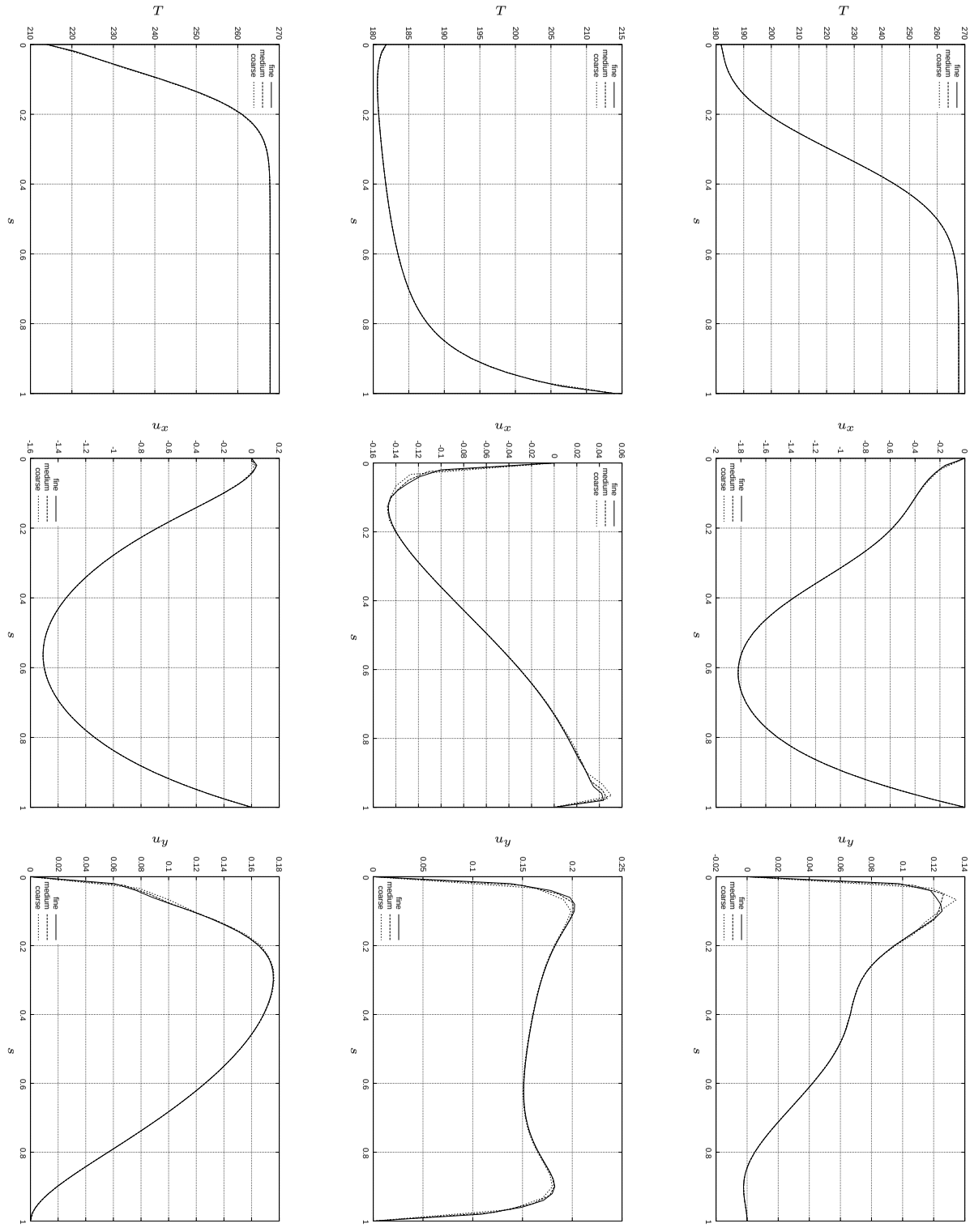


Figure 11.5.3: Tee junction. Asymptotic pressure-based compressible solver. Solution on the section $P_{23}P_{33}$ (left), $P_{22}P_{23}$ (middle) and $P_{22}P_{32}$ (right). s is the curvilinear abscissa. All the quantities are expressed in the SI system.

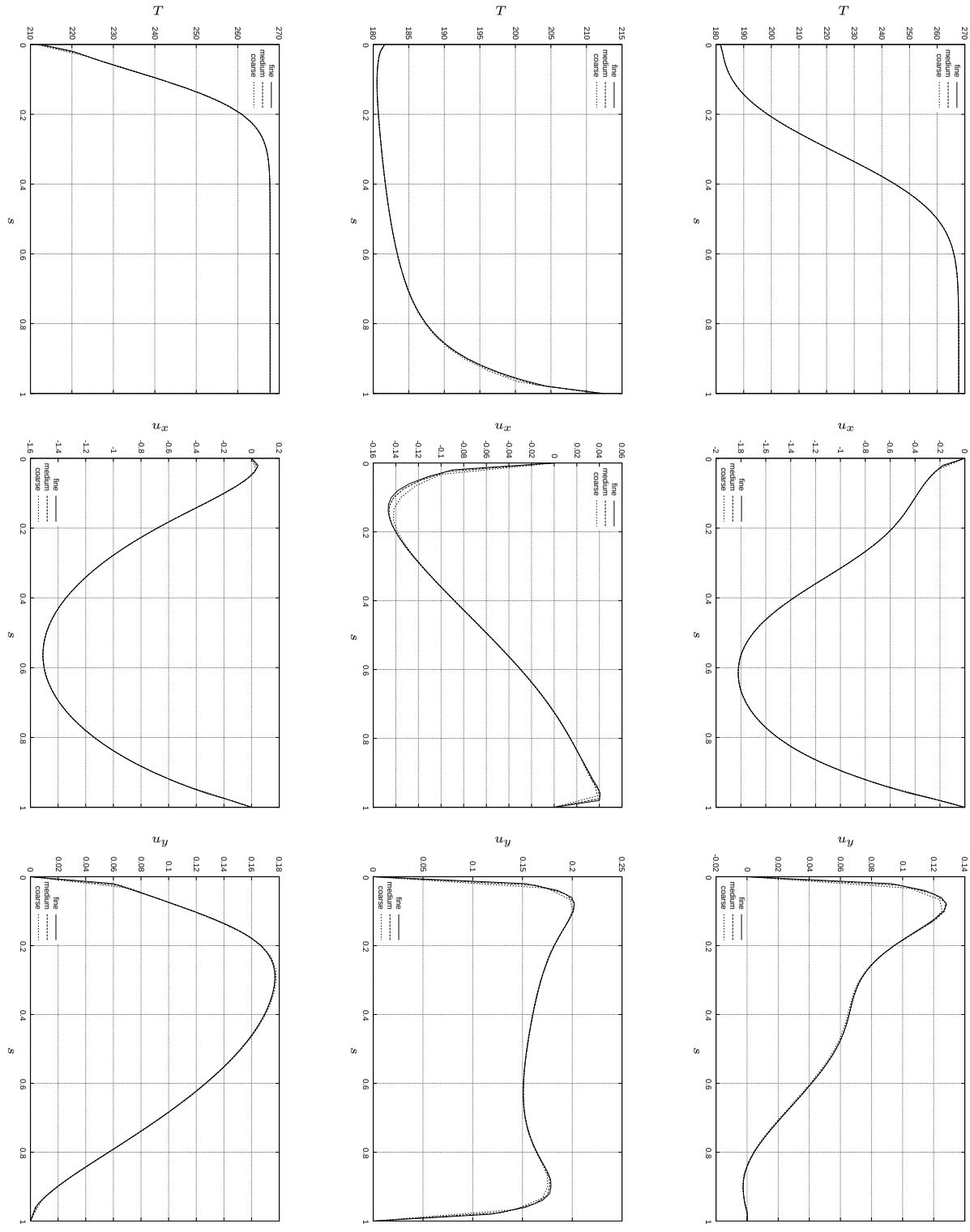


Figure 11.5.4: Tee junction. Density-based compressible solver. Solution on the section $P_{23}P_{33}$ (left), $P_{22}P_{23}$ (middle) and $P_{22}P_{32}$ (right). s is the curvilinear abscissa. All the quantities are expressed in the SI system.

11.5.2 Asymptotic pressure-based solver vs density-based solver

Accuracy

In figures 11.5.5 we represents the evolution of the temperature and the velocity on sections $P_{23}P_{33}$, $P_{22}P_{23}$ and $P_{22}P_{32}$ that we obtained with both approaches. As one can notice, the accuracy provided by the preconditioned density-based solver coupled with the Matrix-Free method is very similar to the one obtained with the asyptotic pressure-based solver.

Performance

All the computations has been carried out on Pentium 4, 2 GHz CPU with 2 GB memory. As mentioned earlier, the number of elements for each method cannot be compared since it does not involve the same degrees of freedom in each case. Typically, the number of elements of the asymptotic method has to be multiplied by 4 to approximate the real number of computation points. In table 11.5.1, we report this approximate number for every grid, in order to perform the comparisons.

Mesh	NBEL	Method	Iterations	CPU	CPI	CPPPI
Coarse	12540	Asymptotic	**180**	4185 s.	23	$185 \cdot 10^{-5}$
	12630	MF-SGS(15)	1050	**1190 s.**	1.13	**8.97** $\cdot 10^{-5}$
Medium	22320	Asymtotic	**220**	7775 s.	35	$158 \cdot 10^{-5}$
	22400	MF-SGS(15)	1380	**2770 s.**	2	**8.96** $\cdot 10^{-5}$
Fine	34900	Asymptotic	**260**	13000 s.	50	$140 \cdot 10^{-5}$
	35000	MF-SGS(15)	2200	**6930** s.	3.15	**9** $\cdot 10^{-5}$

Table 11.5.1: Tee junction. Stationary flow. NBEL is the number of computation points. "Iterations" denotes the number of non-linear iterations to converge to the steady state. CPU is the CPU time to converge. CPI represents the cost per iteration while CPPPI denotes the Cost per iteration and per number of points.

As one can notice, the asymptotic pressure-based solver is strongly efficient in terms of iterations. However, its cost per iteration and per points (CPPPI) is very large. On the other hand, the Matrix-Free implicit scheme possesses a small CPPPI which enables the method, despite a large amount of non-linear iterations, to be more competitive in terms of CPU time. Hence, the Matrix-Free method appears as a valuable alternative technique to compute complex steady viscous flows with CAST3M.

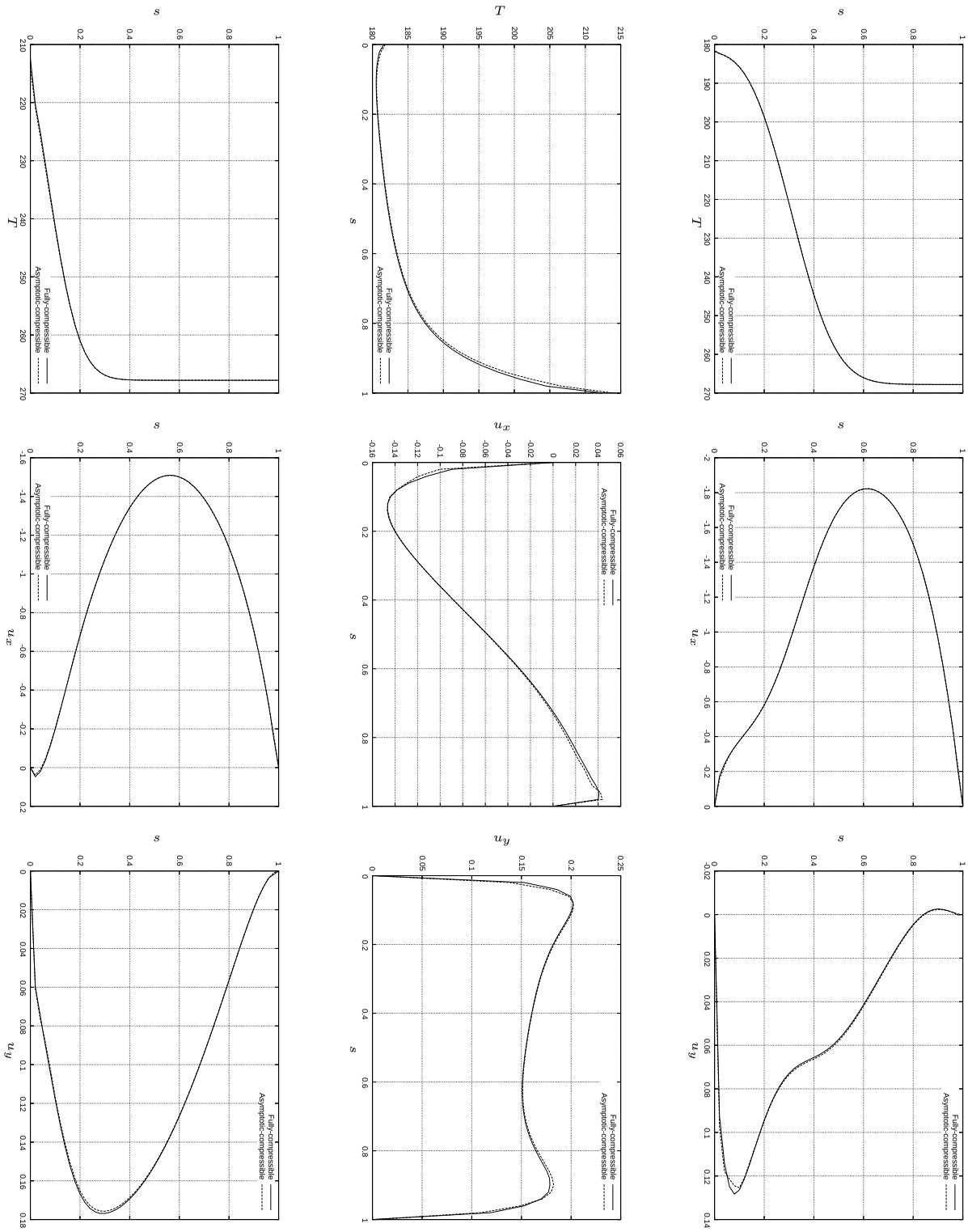


Figure 11.5.5: Tee junction. Compressible solvers. "Fully compressible" refers to the density-based compressible solver. "Asymptotic compressible" refers to the pressure-based asymptotic solver. On the left, solution on the section $P_{23}P_{33}$. In the middle, solution on the section $P_{22}P_{23}$. On the right, solution on the section $P_{22}P_{32}$.

11.6 Non-stationary case

11.6.1 Description of the problem

Let us now study a non-stationary version of the tee junction problem. Starting from the stationary solution previously computed, we modify the boundary conditions for the inlet H. More precisely, we consider the same momentum $\tilde{m}_H$ and we choose a time-dependent temperature $\tilde{T}_H$ as follows :

$$\tilde{T}_H(t) = \tilde{T}_H(\tilde{t} = 0)(1 + A \ \sin(\omega \tilde{t}))$$

where A and ω denote respectively the amplitude and the pulsation of the fluctuations.

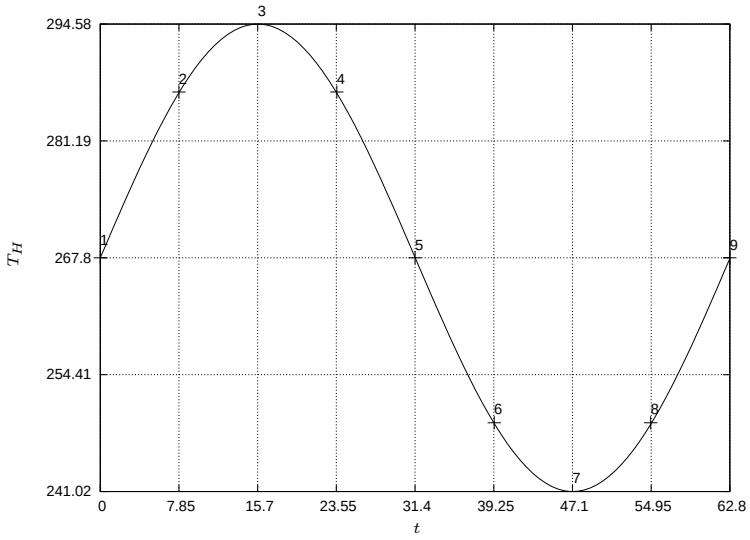


Figure 11.6.1: Tee junction. Non-stationary solution. Evolution of the temperature $\tilde{T}_H$ as function of the time. All quantities are expressed in the SI units.

Different characteristic times are involved in this problem. First, since the convection velocity is about 1 m/s, and the diameter is 1 m, the characteristic time for the temperature convection can be expressed as follows :

$$\tilde{t}_{\text{con,ref}} = \frac{d}{\tilde{\bar{u}}} = 1 \text{ s}$$

Thus, it takes one second for a fluid element to run one diameter. Second, the diffusion coefficient for the temperature reads :

$$\alpha_{\text{ref}} = \frac{\tilde{\lambda}_{\text{ref}}}{\tilde{\bar{\rho}} \ c_p} = 14.4 \cdot \frac{0.4}{1.4 \cdot 288} = 0.0143 \text{ m}^2 \text{ s}^{-1}$$

Hence, the characteristic time for the temperature diffusion can read :

$$\tilde{t}_{\text{dif,ref}} = \frac{d^2}{\alpha_{\text{ref}}} = 70 \text{ s}$$

Eventually, acoustic waves travel at sound speed, namely

$$\tilde{c}_{\text{ref}} = \sqrt{\gamma \frac{\tilde{P}_0}{\tilde{\bar{\rho}}}} = 300 \text{ m s}^{-1} \ .$$

The characteristic time for acoustic wave is then the following :

$$\tilde{t}_{\text{ac,ref}} = \frac{d}{\tilde{c}_{\text{ref}}} = 3.33 \cdot 10^{-3} \text{ s}$$

Table 11.6.1 resumes these characteristic times and their associated pulsations ($2\pi/\tilde{t}$). In order to observe significant temperature variations in the flow field during one period, we have to choose a pulsation smaller than unity and a non-negligible amplitude. We therefore take the following values :

$$A = 0.1 \ , \quad \omega = 0.1 \text{ s}^{-1}$$

The period of the temperature fluctuation at the inlet H is then $2\pi/0.1 = 62.8$ s (*cf.* figure 11.6.1 for the representation of the temperature $\tilde{T}_H$ during one period).

	$\tilde{t}_{\text{ref}}$ (s)	$2\pi/\tilde{t}_{\text{ref}}$ (Hz)
Acoustic	$3.33 \cdot 10^{-3}$	1890
Convective	1	6.28
Diffusive	70	0.0898

Table 11.6.1: Tee junction. Characteristic times.

11.6.2 Numerical solutions

In figure 11.6.2, we present the temperatures isolines in the hot tube at different times, from $\tilde{t} = 7.85$ s to $\tilde{t} = 31.4$ s. We clearly observe the propagation of the temperature fluctuations. In order to investigate further the effects of these variations, we focus on the section $P_{23}P_{33}$. In figures 11.6.3 and 11.6.4, we draw the velocities $\tilde{u}_x$ and $\tilde{u}_y$, and the temperature $\tilde{T}$ for every time level described in figure 11.6.1. At $\tilde{t} = 7.85$ s (label 2), we observe a large variation of the velocity $\tilde{u}_x$ whereas the fluid elements coming from the hot inlet have not reached the section (*cf.* figure 11.6.2 top-left frame). Such a variation can be explained as follows. In fact, as the inlet momentum $\tilde{m}_H$ remains constant with time, the inlet velocity varies with the temperature as follows :

$$\tilde{u}_{x,H} = \frac{\tilde{m}_H}{\tilde{P}} R \tilde{T}_H \ .$$

Besides, according to asymptotic analysis, the speed divergence is subjected to the following condition :

$$\vec{\tilde{\nabla}} \cdot \vec{\tilde{u}} = \frac{\gamma - 1}{\gamma \tilde{P}} \vec{\tilde{\nabla}} \cdot \left(\tilde{\lambda} \vec{\tilde{\nabla}} \tilde{T} \right) \ .$$

In the hot tube, that is between $P_{23}P_{33}$ and $P_{24}P_{34}$, the RHS remains small such that $\vec{\tilde{\nabla}} \cdot \vec{\tilde{u}} \approx 0$. This elliptic constraint is responsible for the fact that the variation of the inlet velocity $\tilde{u}_{x,H}$

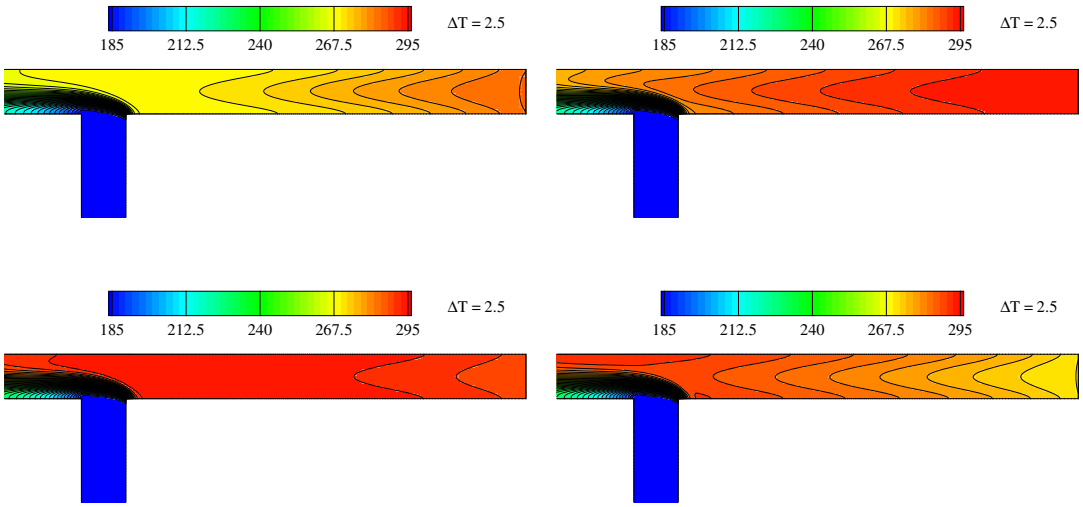


Figure 11.6.2: Tee junction. Non-stationary solution. Fully-compressible solver. Temperature isolines (°K) in the hot tube at $\tilde{t} = 7.85$ s (top-left), at $\tilde{t} = 15.7$ s (top-right), at $\tilde{t} = 23.55$ s (bottom-left) and at $\tilde{t} = 31.4$ s (bottom-right).

immediately affects the speed $\tilde{u}_x$ in section $P_{23}P_{33}$. On the other hand, the temperature does not vary much. This is due to the fact that, as mentioned earlier, the fluid elements arriving from the hot inlet have not already reached the section. As the temperature remains constant while the local velocity $\tilde{u}_x$ increases, the local momentum on the section increases. This reduces the impact of the cold fluid and, as a result, it diminishes the $\tilde{u}_y$ speed.

Let us now compare the solutions at $\tilde{t} = 7.85$ s (label 2) and at $\tilde{t} = 15.7$ s (label 3). As the fluid elements coming from the hot inlet have reached the section (*cf.* figure 11.6.2 top right frame), the variation of the temperature is very important. Notice that the increase of temperature is higher in the middle of the tube where the speed is the largest, *i.e.* where the heat exchanges by convection are the most important. Besides, the speed along x has increased as well as its y-component. The increase of $\tilde{u}_y$ is related to the decrease of the momentum $\tilde{m}_x$ with respect to its value at $\tilde{t} = 7.85$ s, and also to the fact the the buoyancy force has increased because the temperature increases.

From $\tilde{t} = 15.7$ s to $\tilde{t} = 23.6$ s (label 3 and 4 in figure 11.6.3), the velocity $\tilde{u}_x$ decreases with time as expected. Nevertheless, because of the time delay linked to the fluid transport, the temperature in the section keeps on increasing.

From $\tilde{t} = 23.6$ s to $\tilde{t} = 31.4$ s, the temperature decreases. Once again, it decreases more at the tube center than at the tube wall, where the speed is more important, *i.e.* where the heat exchange by convection is more important.

At the other time levels, the behavior is qualitatively the same, namely the velocity $\tilde{u}_x$ immediately follows the variation of the temperature (and of the speed) at the hot inlet; the temperature is affected by a delay time due to convection, while the velocity $\tilde{u}_y$ follows both temperature and momentum variations.

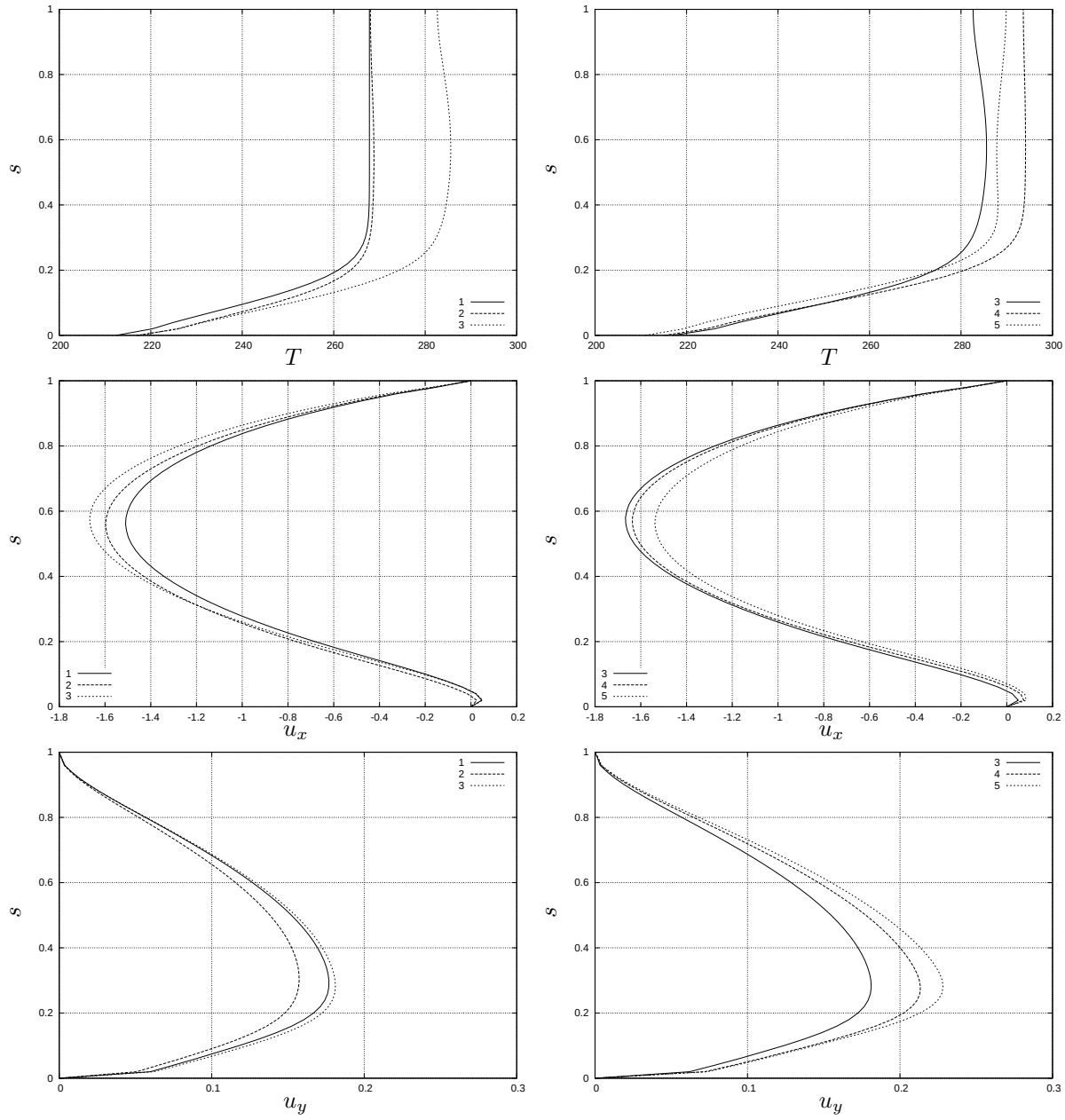


Figure 11.6.3: Tee junction. Non-stationary solution. Evolution of the temperature and velocity on the section $P_{23}P_{33}$ at different times (label 1 refers to the initial time, label 5 to half-period time, as one can see in figure 11.6.1). All quantities are expressed in the SI units.

11.6.3 Density-based solver vs asymptotic pressure-based solver

Accuracy

In order to obtain the same accuracy with both solvers, convergence analyses with respect to the mesh dimension and the time-step are performed. For both methods, computations showed that the fine grid used for the steady case was sufficient.

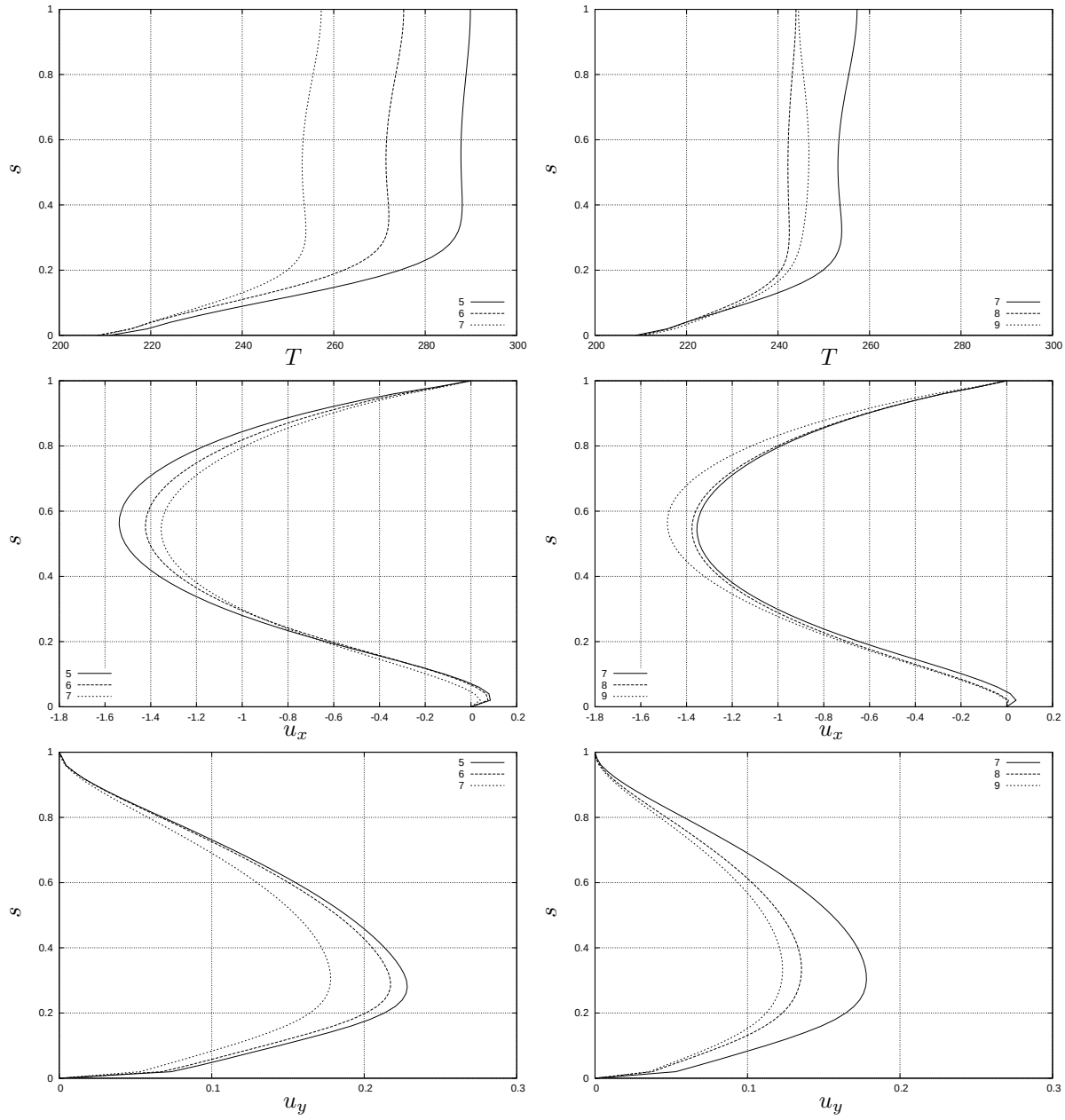


Figure 11.6.4: Tee junction. Non-stationary solution. Evolution of the temperature and velocity on the section $P_{23}P_{33}$ at different times (label 5 refers to half-period time and label 9 to the final time, as one can see in figure 11.6.1). All quantities are expressed in the SI units.

For unsteady flows governed by the particle wave speeds, it was shown in chapter 8 that the particle CFL number (*i.e.* $CFL_u = u\Delta t/\delta x$) should be of order unity (or less) so as to ensure temporal accuracy. From the CPU time point of view, however, it is customary to resort to greater values of this parameter. In practice, great time-step are first choosen then their value is decreased until the numerical solution does not vary much. Thus, the optimal time-steps for

both approaches were likely established. These time-steps were defined as follows :

$$\Delta t = \frac{\pi}{\omega} \cdot \frac{1}{N}$$

where N is an integer. It was found that for the asymptotic pressure-based solver the optimal value for N was 80 while for the density-based solver the optimum was 160. Comparisons between both methods are presented in figures 11.6.5 and 11.6.6. The temperature and the velocities in section $P_{23}P_{33}$ are plotted for the following time-levels : $\tilde{t} = 15.7$ s and $\tilde{t} = 31.4$ s (figure 11.6.5), $\tilde{t} = 47.1$ s and $\tilde{t} = 62.8$ s (figure 11.6.6). As one can notice, the density-based solver coupled with the low-Mach precondioning provides a solution as precise as the one computed using the asymptotic pressure-based approach specially devoted to this type of unsteady low-Mach number flows.

Performances

All the computations were performed using Intel Xeon 3GHz with 2GM Ram memory. As far as the asymptotic pressure-base solver is concerned, we recall that the linear system derived from the Finite-Element space-discretization and the BDF2 time-discretization, is solved using a Picard type fixed-point scheme. The convergence criterion between two physical time levels is based on the loss of five orders for the temperature residual during the iterative process.
In the case of the density-based solver written in a dual-time stepping framework and solved by the MF-SGS scheme, the conditions of the computation are a little bit more complex. Indeed, setting an appropriate value for the unsteady cut-off appeared to be quite difficult. In chapters 7 and 8, a new definition of this parameter was exhibited and it was shown that it provided a good convergence rate for both acoustic and particle problems. This unsteady cut-of is here recalled :

$$\beta_u = \frac{\delta x}{\Delta t \; c} \cdot \frac{\sqrt{CFL_{u+c}}}{\pi}$$

Unfortunately, numerical experiments emphasized that for such a definition the dual-time stepping scheme suffered from a lack of efficiency and robustness. On the other hand, choosing a greater value for this parameter (about ten times the initial value) ensured robustness and enabled to loose 3 orders for the energy residual. However, a precise analysis of the results outlined significant differences between both methods for the u_y velocity in the section $P_{23}P_{33}$ (*cf.* figure 11.6.7).
The exact values of the cut-off were precisely examined since this latter governs not only the convergence rate but also the accuracy of the solution. Indeed, as it was mentionned in chapters 7 and 8, and as it was brought up in [62], a too big cut-off scales unproperly the numerical dissipation leading to an inaccurate solution. The studies revealed that in our case, the modified unsteady cut-off was thirty times as much as the reference Mach number of the problem. Consequently, so as to ensure both robustness and accuracy, we opted for the following strategy : first, we set a great value for the cut-off then we diminish it during the dual-time convergence process until reaching a value of the order of the Mach number. Thus, it was likely to maintain a good convergence rate and to recover the same accuracy than with the pressure-based solver. Nonetheless, it is necessary to outline that such a strategy provides fluctuating performances with respect to the problem which is studied. Moreover, it is not always possible to reduce the cut-off so as to scale properly the dissipation without deteriorate the robustness of the scheme. A likely solution would be to separate the preconditioning formulation used for the time-derivative (which

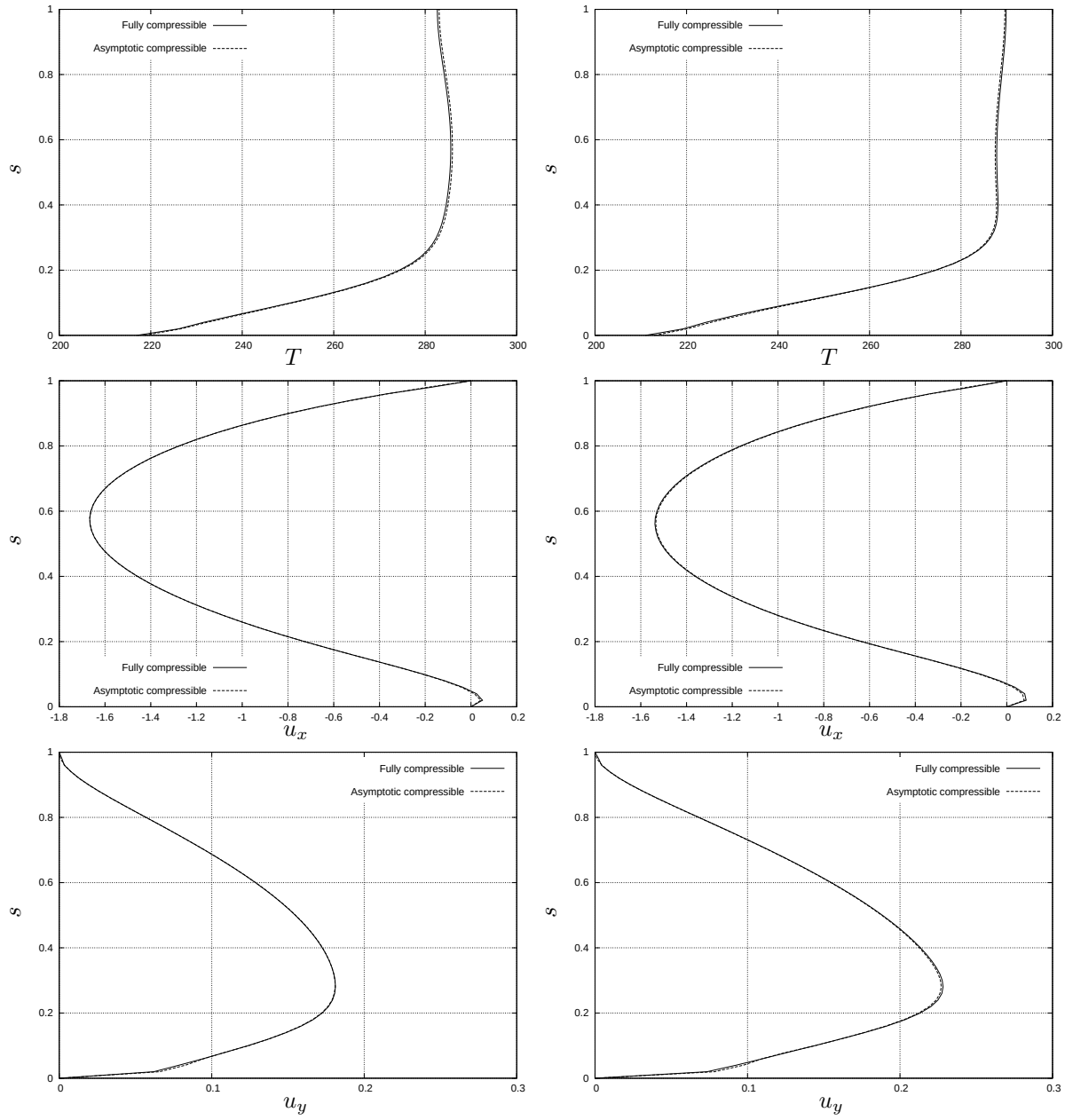


Figure 11.6.5: Tee junction. Non-stationary solution. Comparison between the presure-based and the density based-solvers. Temperature (top frames), $\tilde{u}_x$ velocity (middle frames) and $\tilde{u}_y$ velocity (bottom frames) are represented on section $P_{23}P_{33}$ and for the following time levels : $\tilde{t} = 15.7$ s and $\tilde{t} = 31.4$ s. All quantities are expressed in the SI units.

is responsible for convergence efficiency) and that used for the artificial dissipation terms (responsible for accuracy). Such a formulation may be implemented within a multiple pseudo-time framework as recently discussed in [82]. Implementation of such a multi-level preconditioning formulation will have to be the subject of future work.

Despite a certain loss of efficiency with respect to the steady case, the density-based solver coupled with the MF-SGS implicit treatment provides an accurate solution for a reasonable

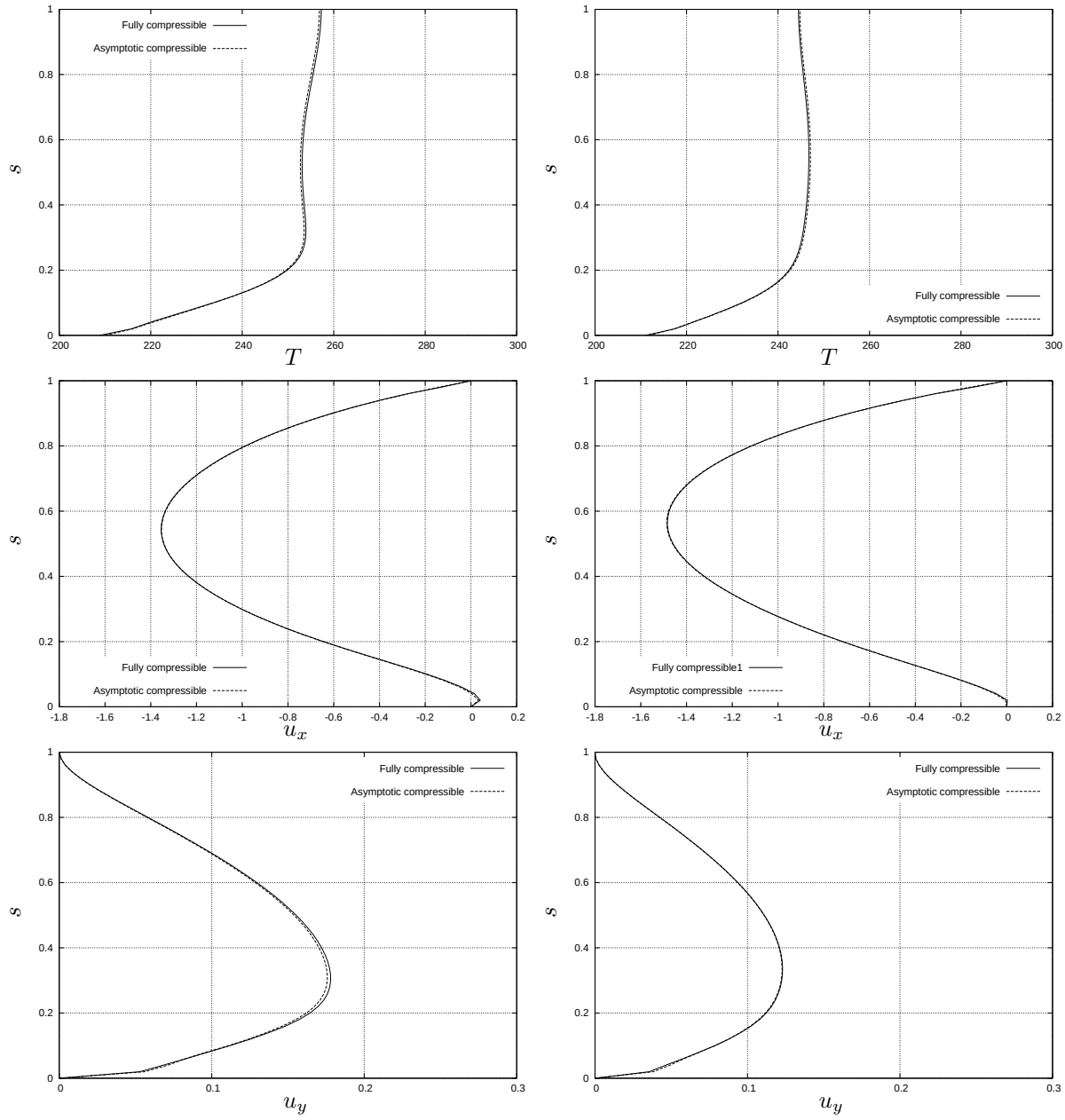


Figure 11.6.6: Tee junction. Non-stationary solution. Comparison between the presure-based and the density based-solvers. Temperature (top frames), $\tilde{u}_x$ velocity (middle frames) and $\tilde{u}_y$ velocity (bottom frames) are represented on section $P_{23}P_{33}$ and for the following time levels : $\tilde{t} = 47.1$ s and $\tilde{t} = 62.8$ s. All quantities are expressed in the SI units.

computational cost as one can see in the summarizing table 11.6.2. As mentioned previously, the single dual-time stepping formulation cannot enable to define a versatile unsteady cut-off in order to ensure both efficiency and accuracy. Moreover, the density-based solver requires twice as many physical-time iterations as its pressure-based counterpart which penalizes significantly the global computational cost. The reasons of the need of a smaller physical time-step have not been clearly established yet.

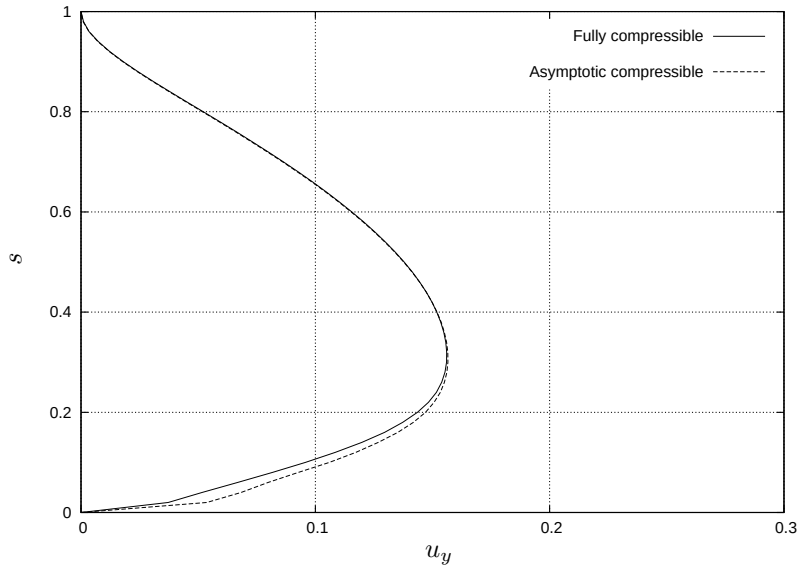


Figure 11.6.7: Tee Junction. Non-stationnary solution. Comparison between the density-based solver and the pressure-based solver. Curves of the velocity $\tilde{u}_y$ in section $P_{23}P_{33}$ at time level $\tilde{t} = 15.7$ s for a big value of the unsteady cut-off. All the quantities are expressed using SI units.

Mesh	NBEL	Methode	CPU
Fine	34900	Asymptotic	**27000** s.
	35000	MF-SGS(30)	36500 s.

Table 11.6.2: Tee Junction. Unsteady flow. NBEL is the number of computation points. CPU is the global compuattion cost.

11.7 Conclusions

This last chapter enabled to validate the Matrix-Free implicit method over an application of interest for the LTMF laboratory, namely the modelling of Tee junction at low-Mach number, for steady and unsteady regimes. Performances of the Matrix-Free implicit scheme both in terms of accuracy and computational cost were compared with the asymptotic pressure-based solver available in CAST3M.

In the case of the low-Mach number steady flow, the newly implemented Matrix-Free implicit scheme ensures the same level of accuracy than its asymptotic pressure-based couterpart. Moreover, for a same number of computational points, it provides a significant gain of CPU time. The Matrix-Free implicit treatment can be regarded as a genuine competitive alternative for the computation of viscous low-Mach number steady flows.

On the other hand, the preconditioned dual-time stepping implicit scheme is less competitive to deal with the low-Mach number unsteady flows. The loss of efficiency is mainly related to the difficulty of defining an unsteady cut-off that ensures both good damping and accuracy. Indeed, when the value of this parameter is widely greater than the local Mach number, a fast convergence is obtained at the expense of a lack of accuracy. Conversely, when the cut-off is of the order of the local Mach number, accuracy is preserved but the scheme then suffers from a lack of robustness and when it remains stable its convergence rate becomes poor. Moreover, it appears that the Matrix-Free implicit framework is more sensitive to this kind of instabilities than its block counterpart. However, this difficulty might be circumvented through using separate preconditioners for the dissipation and pseudo-time derivative terms. The former then controls the accuracy of the discrete formulation, while the latter controls convergence efficiency of the sub-iterations. This technique has already been developed in [82] via a multi-level preconditioning formulation and will be certainly implemented in the future. However, in spite of its lower efficiency, the preconditioned density-based solver coupled with the Matrix-Free implicit scheme is still attractive for the simulation of unsteady flows since, unlike its pressure-based couterpart, it can handle with a wide range of speeds for reasonable computational cost and memory storage.

Conclusions

The present thesis was motivated by a demand from the CEA for more efficient implicit treatments in the context of unsteady low-Mach number flows, computed using preconditioned density-based upwind schemes. The existing implicit treatments in CEA's numerical platform CAST3M were oriented towards the maximization of intrinsic efficiency at the expense of a high unit computational cost and, most of all, stringent memory requirements that were becoming prohibitive when large-scale computations were undertaken. The choice was made in this work to target an implicit treatment favoring the minimization of the unit computational cost and the minimization of memory requirements. Naturally, it was admitted from the start that such a method would provide a poor intrinsic efficiency and thus require a large number of iterations to reach a steady-state (be it in physical or dual time); however the objective was to reduce the unit cost of the method in such a way it would balance anyhow the lack of intrinsic efficiency and allow the method to remain competitive in terms of global computational cost with respect to the existing treatment, while offering modest memory constraints.
In the course of this work, it appeared rapidly a so-called matrix-free approach was a good candidate for being retained as a low-cost alternative implicit treatment in CAST3M. Some issues had to be solved however :

- it was imperative to make the approach attractive for the whole Mach number range but the use of a preconditioning matrix was likely to increase the unit cost of the method by making it lose its truly matrix-free character. Taking advantage of the fact Turkel's preconditioning matrix is idempotent allowed to fix this problem since the "matrix-free" treatment can then be expressed under a form that remains cheap to compute even when the preconditioning is on; consequently, the matrix-free treatment remains globally efficient for all-speed flows.

- the matrix-free treatment as such defines an implicit stage that must be solved using some kind of (iterative) solution technique. The choice of an efficient solution technique was constrained by the need to use the method in a unstructured grid context, which ruled out for instance any line-relaxation technique. An *a priori* Von Neumann analysis allowed to select a Symmetric Gauss-Seidel (SGS) approach as a better choice over a basic Point Jacobi (PJ) technique. However, the efficiency of the SGS procedure relies on the ordering of the mesh cells and, moreover, for 3D computations, it requires more connectivity information than the PJ approach. At any case, both approaches are easy to implement within an unstructured framework and enable to reduce dramatically the memory storage of the implicit treatment.

- it appeared also necessary to adapt the definition of the low-Mach cut-off in order to ensure the efficiency of the Matrix-Free method for solving unsteady viscous low-Mach number

flows. The Von Neumann analysis of the amplification matrix for the linearized Euler and Navier-Stokes equations enabled to design and calibrate these cut-off values which play a crucial role in the performance of preconditioned upwind schemes. Some choices were then proposed for these values, that proved to perform well on real-life applications.

The assessment of the proposed matrix-free treatment for all-speed flows has been carried out in two stages : a preliminary set of validations for a large spectrum of test-cases (inviscid and viscous, steady and unsteady, low-Mach and compressible flows) has been computed using a structured flow solver specifically tailored for this sole purpose. The results obtained in terms of global efficiency and memory requirement were convincing enough to undertake an implementation within the general-purpose CAST3M code. The results obtained from a comparative study between the existing implicit treatment (block-implicit stage solved by a Newton-Krylov approach) and the low-cost approach demonstrate the interest of the matrix-free treatment both in terms of global computational cost and memory requirement. The computation of a low-Mach number unsteady viscous flow in a tee configuration presented in the last chapter of this thesis shows in particular the preconditioned implicit density-based solver is a genuine alternative to an existing pressure-based solver of a low-Mach asymptotic model.
However, some remaining unsolved issues deserve to be pointed out and the developments yet to perform in order to make our tool applicable to the targeted hydrogen combustion problems mentioned in the introduction of this thesis must be recalled :

- the matrix-free treatment presented in this thesis has also been applied to internal flow problems, such as the differentially heated square cavity or an injection problem [61]. It appears that, for this type of problems, the efficiency of the low-cost treatment degrades in such a proportion the method eventually becomes (much) less efficient than the existing block-implicit approach. It is postulated this lack of efficiency can be related to the treatment of boundary conditions but further studies are strongly needed in order to make the low-cost treatment globally efficient for such configurations.

- only a single-species version of the free-matrix treatment has been implemented within CAST3M. The (truly straightforward) extension of the approach to multi-species flow problems is of course needed for tackling hydrogen combustion problems and such a task should probably be the first step to take following the conclusion of the present thesis.

- eventually, we want to stress that it is still a challenging task to apply preconditioned compressible upwind solvers to the resolution of unsteady low-Mach number flows. Indeed, the new definition of the unsteady cut-off designed for the Matrix-Free implicit treatment cannot ensure simultaneously efficiency and accuracy demands. Thus, elaborate strategies have to be developed for each case so as to unify these both requirements. A likely solution to circumvent such issues, would be to separate the preconditioning formulation used for the time-derivative (which is responsible for convergence efficiency) and that used for the artificial dissipation terms (which control accuracy). Such a promising formulation may be implemented within a multiple pseudo-time framework [82] and would be certainly the subject of future developments.

Appendix

Appendix A

Matrices of the Navier-Stokes System

A.1 Euler Equations

A.1.1 Jacobian Matrices for the inviscid fluxes [31]

The dimensionless system of the Euler equations can be written under the following form :

$$w_t \; + \; (f^E(w))_x \; + \; (g^E(w))_y \; = \; 0 \tag{A.1.1}$$

where t is the dimesionless time, x and y are the spacial coordinates, $w = w(x, y, t)$ is the vector of conserved variables, and f^E and g^E are the inviscid flux vector functions,

$$w \; = \; \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} \quad , \quad f^E \; = \; \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix} \quad , \quad g^E \; = \; \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{pmatrix} \quad . \tag{A.1.2}$$

where all the dimensionless quantities have been already defined in the first chapter. The system is closed by the dimensionless equation of state :

$$p \; = \; \rho T \; = \; (\gamma - 1)\rho \left(E - \frac{1}{2}|\vec{u}|^2 \right) \; = \; (\gamma - 1)\rho e \quad . \tag{A.1.3}$$

The dimensionless sound speed is defined as follows :

$$c = \sqrt{\gamma \; T} = \sqrt{\frac{\gamma \; p}{\rho}}$$

The specific total energy and enthalpy can then be defined as :

$$E = \frac{c^2}{\gamma(\gamma - 1)} + \frac{1}{2}(u^2 + v^2) \quad , \qquad H = \frac{c^2}{(\gamma - 1)} + \frac{1}{2}(u^2 + v^2) \quad .$$

Now, we can easily compute the inviscid Jacobian matrices :

$$A^E = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{\gamma-3}{2}u^2 + \frac{\gamma-1}{2}v^2 & (3-\gamma)u & -(\gamma-1)v & \gamma-1 \\ -uv & v & u & 0 \\ -\gamma u E + (\gamma-1)u(u^2+v^2) & \gamma E - \frac{\gamma-1}{2}(v^2+3u^2) & -(\gamma-1)uv & \gamma u \end{pmatrix} \quad \text{(A.1.4)}$$

$$B^E = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -uv & v & u & 0 \\ \frac{\gamma-3}{2}v^2 + \frac{\gamma-1}{2}u^2 & -(\gamma-1)u & (3-\gamma)v & \gamma-1 \\ -\gamma v E + (\gamma-1)v(u^2+v^2) & -(\gamma-1)uv & \gamma E - \frac{\gamma-1}{2}(u^2+3v^2) & \gamma v \end{pmatrix} \quad \text{(A.1.5)}$$

A.1.2 Transformation matrices [31]

In order to obtain the eigensystem derived from the Euler equations, we need the transformation matrices such that the Jacobian A^E is diagonalized as follows :

$$P_A^{-1} \cdot A^E \cdot P_A = \begin{pmatrix} u & & & \\ \cdot & u & & \\ \cdot & \cdot & u+c & \\ \cdot & \cdot & \cdot & u-c \end{pmatrix}$$

The general expression of these transformation matrices is given by :

$$P_\kappa = \begin{pmatrix} 1 & 0 & \frac{\rho}{2c} & \frac{\rho}{2c} \\ u & \rho\hat{\kappa}_y & \frac{\rho}{2c}(u + c\hat{\kappa}_x) & \frac{\rho}{2c}(u - c\hat{\kappa}_x) \\ v & -\rho\hat{\kappa}_x & \frac{\rho}{2c}(v + c\hat{\kappa}_y) & \frac{\rho}{2c}(v - c\hat{\kappa}_y) \\ \frac{u^2+v^2}{2} & \rho(u\hat{\kappa}_y + v\hat{\kappa}_x) & \frac{\rho}{2c}(H + c\vec{V}\cdot\vec{1}_\kappa) & \frac{\rho}{2c}(H - c\vec{V}\cdot\vec{1}_\kappa) \end{pmatrix}$$

and

$$P_\kappa^{-1} = \begin{pmatrix} 1 - \frac{\gamma-1}{2}M^2 & (\gamma-1)\frac{u}{c^2} & (\gamma-1)\frac{v}{c^2} & -\frac{\gamma-1}{c^2} \\ \frac{1}{\rho}(v\hat{\kappa}_x - u\hat{\kappa}_y) & \frac{\kappa_y}{\rho} & -\frac{\kappa_x}{\rho} & 0 \\ \frac{c}{\rho}(\frac{\gamma-1}{2}M^2 - \frac{\vec{V}\cdot\vec{1}_\kappa}{c}) & \frac{1}{\rho}(\hat{\kappa}_x - (\gamma-1)\frac{u}{c}) & \frac{1}{\rho}(\hat{\kappa}_y - (\gamma-1)\frac{v}{c}) & \frac{\gamma-1}{\rho c} \\ \frac{c}{\rho}(\frac{\gamma-1}{2}M^2 + \frac{\vec{V}\cdot\vec{1}_\kappa}{c}) & -\frac{1}{\rho}(\hat{\kappa}_x + (\gamma-1)\frac{u}{c}) & -\frac{1}{\rho}(\hat{\kappa}_y + (\gamma-1)\frac{v}{c}) & \frac{\gamma-1}{\rho c} \end{pmatrix}$$

When $\kappa = A$ then $\hat{\kappa}_x = 1$, $\hat{\kappa}_y = 0$ and $\vec{1}_\kappa = \vec{1}_x$. On the other hand, when $\kappa = B$ then $\hat{\kappa}_x = 0$, $\hat{\kappa}_y = 1$ and $\vec{1}_\kappa = \vec{1}_y$.

A.2 Navier-Stokes equations

A.2.1 Viscous Jacobian matrices

The dimensionless system of the Navier-Stokes equations can be written under the following form :

$$w_t \; + \; (f^E(w) - f^V(w, w_x, w_y))_x \; + \; (g^E(w) - g^V(w, w_x, w_y))_y \; = \; 0 \qquad \text{(A.2.1)}$$

where f^V and g^V denote the viscous fluxes,

$$f^V \; = \; \frac{1}{Re} \begin{pmatrix} 0 \\ \tau_{xx} = \frac{4}{3}\mu u_x - \frac{2}{3}\mu v_y \\ \tau_{xy} = \mu(u_y + v_x) \\ u\tau_{xx} + v\tau_{xy} + \frac{\gamma\mu}{Pr} e_x \end{pmatrix} \quad , \quad g^V \; = \; \frac{1}{Re} \begin{pmatrix} 0 \\ \tau_{yx} = \mu(u_y + v_x) \\ \tau_{yy} = \frac{4}{3}\mu v_y - \frac{2}{3}\mu u_x \\ u\tau_{yx} + v\tau_{yy} + \frac{\gamma\mu}{Pr} e_y \end{pmatrix} \quad . \qquad \text{(A.2.2)}$$

All the dimensionless quantities are defined in Chapter 1. The viscous Jacobian matrices are expressed as follows :

$$A_0^V(w, w_x, w_y) = \frac{\partial f^V}{\partial w}(w, w_x, w_y) \quad ; \qquad B_0^V(w, w_x, w_y) = \frac{\partial g^V}{\partial w}(w, w_x, w_y)$$

$$A_1^V(w, w_x, w_y) = \frac{\partial f^V}{\partial w_x}(w, w_x, w_y) \quad ; \qquad B_1^V(w, w_x, w_y) = \frac{\partial g^V}{\partial w_x}(w, w_x, w_y)$$

$$A_2^V(w, w_x, w_y) = \frac{\partial f^V}{\partial w_y}(w, w_x, w_y) \quad ; \qquad B_2^V(w, w_x, w_y) = \frac{\partial g^V}{\partial w_y}(w, w_x, w_y)$$

In order to ensure the diagonal dominance of the implicit upwind scheme, we neglected the contribution of the tangential Jacobians in the implicit operator (*cf.* Chapter 1), more precisely in the x-direction we only retained the Jacobian A_1^V while in the y-direction we retained B_2^V (see also [16]). These Jacobian matrices can be written as follows :

$$A_1^V = \frac{\mu}{Re\ \rho}\ \bar{A}_1^V = \frac{\mu}{Re\ \rho} \begin{pmatrix} 0 & 0 & 0 & 0 \\ -\frac{4}{3}u & \frac{4}{3} & 0 & 0 \\ -v & 0 & 1 & 0 \\ -\alpha u^2 - \beta v^2 - \frac{\gamma}{Pr}E & \alpha u & \beta v & \frac{\gamma}{Pr} \end{pmatrix}$$

$$B_2^V = \frac{\mu}{Re\ \rho}\ \bar{B}_2^V = \frac{\mu}{Re\ \rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -u & 1 & 0 & 0 \\ -\frac{4}{3}v & 0 & \frac{4}{3} & 0 \\ -\alpha v^2 - \beta u^2 - \frac{\gamma}{Pr}E & \beta u & \alpha v & \frac{\gamma}{Pr} \end{pmatrix}$$

with

$$\alpha = \frac{4}{3} - \frac{\gamma}{Pr} \qquad \text{and} \qquad \beta = 1 - \frac{\gamma}{Pr}$$

To define the explicit operator, we also need the matrix S^V which is given by :

$$S^V = A_2^V + B_1^V = \frac{\mu}{3Re\ \rho}\ \bar{S}^V = \frac{\mu}{3Re\ \rho}\begin{pmatrix} 0 & 0 & 0 & 0 \\ -v & 0 & 1 & 0 \\ -u & 1 & 0 & 0 \\ -2uv & v & u & 0 \end{pmatrix}$$

A.2.2 Viscous spectral radius

As A_1^V and B_2^V are lower triangular matrices, their spectral radius can be computed straightforwardly. Moreover they share the same diagonal terms, involving that their respective spectral radius are equal. This spectral radius is given by :

$$\rho^V = \frac{\mu}{Re\ \rho} \cdot \max\left[\frac{\gamma}{Pr}, \frac{4}{3}\right]$$

For the air at moderate temperature, we then have :

$$\rho^V = \frac{\mu}{Re\ \rho} \cdot \frac{\gamma}{Pr}$$

Appendix B

Von Neumann Analysis

B.1 Introduction

Several methods have been developed in order to study the efficiency of numerical schemes. In most of these methods, one neglects the effect of the boundary conditions by assuming that the computational domain is limitless or by using periodic boundary conditions. The Von Neumann method used in this work to study the stability of several numerical schemes is based on the Fourier transform and the spectral analysis of these schemes. In practice, one checks that the value of the amplification factor of the schemes remains smaller than unity on the whole domain. The Von Neumann analysis enables also to estimate how fast the error will be damped (or amplified), providing an idea of how the convergence rate of the algorithm evolves. Of cource, the results obtained by such a method have to be commented carefully. Indeed, the Von Neumann theory requires a linear problem which means that the coefficients of the equations are constant and, as mention earlier, periodic boundary conditions are assumed which enables the use of Fourier series. In spite of these limitations, stability results were shown to be meaningful for practical computations. Besides the Von Neumann method can be used very easily through a computer program in order to lead systematic studies.

B.2 Basic principles

In order to understand the fundamental principles of the Von Neumann analysis, let us consider the linear scalar advection problem.

B.2.1 Linear scalar advection equation

$$\frac{\partial u}{\partial t} + a \cdot \frac{\partial u}{\partial x} = 0 \tag{B.2.1}$$

Let $\bar{u}$ be the exact solution of the equation (B.2.1), a discrete solution of this equation reads :

$$u_i^n \;=\; \bar{u}_i^n \;+\; \epsilon_i^n \tag{B.2.2}$$

where ϵ_i^n denotes the error at time level $t_n = n \cdot \Delta t$ and at mesh point $x_i = i \cdot \delta x$. Thus, the centered scheme reads :

$$\frac{\bar{u}_i^{n+1} - \bar{u}_i^n}{\Delta t} + \frac{\epsilon_i^{n+1} - \epsilon_i^n}{\Delta t} = -\frac{a}{2\delta x} \cdot (\bar{u}_{i+1}^n - \bar{u}_{i-1}^n) - \frac{a}{2\delta x} \cdot (\epsilon_{i+1}^n - \epsilon_{i-1}^n) \tag{B.2.3}$$

or

$$\frac{\epsilon_i^{n+1} - \epsilon_i^n}{\Delta t} = -\frac{a}{2\delta x} \cdot (\epsilon_{i+1}^n - \epsilon_{i-1}^n) \tag{B.2.4}$$

The error ϵ is also a solution of the discrete advection equation.

B.2.2 Fourier Series

If the boundary conditions are assumed to be periodic, the error ϵ_i^n can be decomposed into a Fourier series in space at each time level. Since the space domain is of a finite length one will have a discrete Fourier representation summed over a finite number of harmonics.

In a one-dimensional domain of length $2L$, the maximum wavelength that can be captured with a mesh (*cf.* figure B.2.1) is :

$$\lambda_{max} = 2L \tag{B.2.5}$$

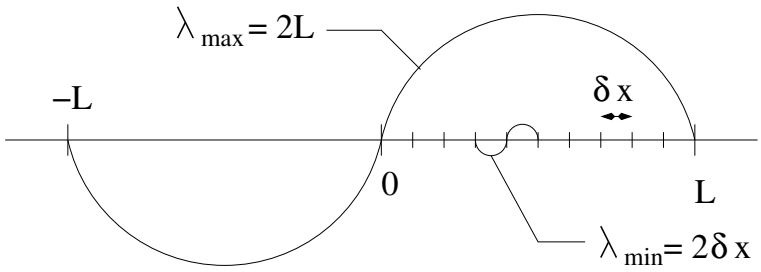


Figure B.2.1: Maximum and minimum wavelengths captured by a finite mesh

On the other hand, the minimum wavelength that can be obtained relies on the size of the mesh δx :

$$\lambda_{min} = 2\delta x \tag{B.2.6}$$

Wavenumbers associated with the wavelengths can be determined as follows :

$$k = \frac{2\pi}{\lambda} \tag{B.2.7}$$

Hence, one gets:

$$k_{min} = \frac{\pi}{L}$$
$$k_{max} = \frac{\pi}{\delta x}$$

Let N be the number of space-steps δx requiered to discretize length L, the harmonics represented on the mesh are given by :

$$k_j \;=\; j \cdot k_{min} \;=\; j \cdot \frac{\pi}{L} \qquad \forall j \in [|0; N|] \tag{B.2.8}$$

The error ϵ_i^n is decomposed into a Fourier series as :

$$\epsilon_i^n = \sum_{j=-N}^{N} E_j^n \exp(\imath \cdot k_j \cdot i\delta x) \tag{B.2.9}$$

Th produce $k_j \cdot \delta x$ denotes a phase angle, or reduced wavenumber, in the Fourier space :

$$\phi \;=\; k_j \cdot \delta x \tag{B.2.10}$$

Thus, the Fourier series read :

$$\epsilon_i^n = \sum_{j=-N}^{N} E_j^n \exp(\imath \cdot i\phi) \tag{B.2.11}$$

The reduced wavenumber covers the domain $[-\pi;\ \pi]$ in steps of π/N :

$$\begin{aligned} \phi_{min} \;&=\; \frac{\delta x}{L} \cdot \pi \\ \phi_{max} \;&=\; \pi \end{aligned}$$

The region around $\phi = 0$ corresponds to low frequencies while the region close to $\phi = \pi$ is associated with the high-frequency of the spectrum. In particular, the value $\phi = \pi$ is associated with the highest frequency resolvable on the mesh, namely the frequency of wavelength $\lambda_{min} = 2\delta x$.

B.2.3 Amplification factor

The problem is assumed to be linear, each harmonic of the error ϵ_i^n is therefore a solution of the discrete equation :

$$\frac{E^{n+1} - E^n}{\Delta t} \cdot \exp(\imath\ i\phi) + \frac{a}{2\delta x} \cdot (E^n \exp(\imath\ (i+1)\phi) - E^n \exp(\imath\ (i-1)\phi)) = 0 \tag{B.2.12}$$

Dividing by $\exp(\imath\ i\phi)$ leads to :

$$E^{n+1} - E^n \;+\; \frac{\sigma}{2}\ E^n(\exp(\imath\ \phi) - \exp(-\imath\ \phi)) = 0 \tag{B.2.13}$$

where $\sigma = \dfrac{a\Delta t}{\delta x}$.

Th stability condition is satisfied if the amplitude of any error harmonic E^n does not grow in time, that is, the ratio

$$|G| \;=\; \left|\frac{E^{n+1}}{E^n}\right| \leq 1 \qquad \forall \phi\ . \tag{B.2.14}$$

G is the amplification factor, that is a function of time-step Δt, reduced wavenumber ϕ and mesh size δx. For the centered scheme, one has :

$$G \;=\; 1 \;-\; \imath\,\sigma\,\sin(\phi) \tag{B.2.15}$$

The modulus of G is, in this case, greater than 1 for all reduced wavenumbers :

$$|G|^2 \;=\; 1 \;+\; \sigma^2\,\sin^2(\phi) \;\geq\; 1 \qquad \forall\phi \tag{B.2.16}$$

Consequently, the centered scheme is unconditionally unstable.

B.3 System of equations

B.3.1 Linearization of the system

Let us consider the hyperbolic system of conservation laws :

$$w_t + f(w)_x = 0 \tag{B.3.1}$$

where $w(x,t)$ is a vector of m components and f is a flux function associated with the Jacobian matrix $A = \partial f/\partial w$. The system (B.3.1) can be approached by a first-order accurate upwind explicit scheme :

$$w_i^{n+1} - w_i^n = -\frac{\Delta t}{\delta x}\left(\delta\mu f_i^n - \frac{1}{2}\,\delta(|A^n|\delta w^n)_i\right) \tag{B.3.2}$$

We recall that the space discretization is achieved using difference operators :

$$\delta[\psi(x)] \;=\; \psi(x+\frac{\delta x}{2}) - \psi(x-\frac{\delta x}{2})\;, \quad \mu[\psi(x)] \;=\; \frac{1}{2}\psi(x+\frac{\delta x}{2}) + \frac{1}{2}\psi(x-\frac{\delta x}{2}) \tag{B.3.3}$$

where ψ is defined at each mesh point $x_i = i\cdot\delta x$ and δx is the constant mesh size. First, the system is linearized by assuming that the Jacobian matrix A is constant. Thus the flux function reads :

$$f(w) = A\cdot w \tag{B.3.4}$$

Second, the linear scheme is made implicit :

$$\forall n\in\mathbf{N} \quad \mathcal{H}\cdot\Delta w^n = -\mathcal{K}\cdot w^n \tag{B.3.5}$$

where $\mathcal{H}$ and $\mathcal{K}$ are linear operators defined as :

$$\forall i\in\mathbf{Z} \quad \begin{cases} \mathcal{H}\cdot\Delta w_i^n = (Id + \dot{A}\;\mu\delta - \dfrac{1}{2}\;|\dot{A}|\;\delta^2)\Delta w_i^n \\ \mathcal{K}\cdot w_i^n = (\dot{A}\;\mu\delta - \dfrac{1}{2}\;|\dot{A}|\;\delta^2) w_i^n \end{cases} \qquad \text{with } \dot{A} = \frac{\Delta t}{\delta x}A \text{ and } \delta^2 = \delta\;\circ\;\delta\;. \tag{B.3.6}$$

In order to study the amplification factor of the implicit upwind scheme associated with the system of conservation laws, one need to use the Fourier transform.

B.3.2 Fourier transform

Since the numerical solution is not necessarily periodic, one uses the Fourier transform. First, one extends the linear scheme (B.3.5) to functions defined not only at the spatial mesh points $x = i \cdot \delta x$, but for all $x \in \mathbf{R}$. For $\psi \in L_2(\mathbf{R})^m$, let $\widehat{\psi} = \Phi(\psi)$ denote the spatial Fourier transform, that is the function of $L_2(\mathbf{R})^m$ defined as :

$$\Phi(\psi) = \widehat{\psi}(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \exp(-\underline{\mathrm{i}}\, \frac{\xi}{\delta x}\, x)\ \psi(x)\ dx \quad . \tag{B.3.7}$$

where ξ denotes the reduced wave number and $\underline{\mathrm{i}}^2 = -1$. For any function $\psi \in L_2(\mathbf{R})^m$, combinations of space difference operators read :

$$(\delta\mu\psi)(x) = \frac{\psi(x+\delta x) - \psi(x-\delta x)}{2} \ ,$$

$$(\delta^2\psi)(x) = \psi(x+\delta x) - 2\psi(x) + \psi(x-\delta x) \ ,$$

Applying the Fourier transform to these functions leads to :

$$\Phi(\delta_1\mu_1\psi)(\xi) = \underline{\mathrm{i}}\ \sin(\xi)\widehat{\psi} \ ,$$

$$\Phi(\delta_1^2\phi)(\xi) = -2(1-\cos(\xi))\widehat{\psi} \ ,$$

Thus, the transformed space difference operators and their combinations take a very simple form. By transforming each side of equation (B.3.5) one obtains :

$$H \cdot \Delta\widehat{w}_i^{n+1} = -K \cdot \widehat{w}_i^{n+1} \quad , \tag{B.3.8}$$

where H, K and $\widehat{w}^{n+1}$ denote the transformed functions of $\mathcal{H}$, $\mathcal{K}$ and w^{n+1} respectively. The operators H and K read :

$$\begin{cases} K &= |\dot{A}|(1-\cos(\xi)) \ + \ \underline{\mathrm{i}}\ \dot{A}\sin(\xi) \\ H &= Id + |\dot{A}|(1-\cos(\xi)) \ + \ \underline{\mathrm{i}}\ \dot{A}\sin(\xi) \end{cases}$$

B.3.3 Amplification factor

The amplication matrix of the linear implicit upwind scheme associated with the system of conservation laws is defined as follows :

$$\forall n \in \mathbf{N} \quad \widehat{w}^{n+1} = G \cdot \widehat{w}^n \tag{B.3.9}$$

In the present case, one gets :

$$G = Id - H^{-1} \cdot K \tag{B.3.10}$$

The Von Neumann stability condition is defined by :

$$\forall\ \xi \in [-\pi,\ \pi] \quad \rho(G(\xi)) \ \leq \ 1 \tag{B.3.11}$$

where $\rho(G)$ is the spectral radius of the amplification matrix G. Given to the large number of parameters that one has to take into account to analyse the stability of such a scheme, a numerical study using sweeps over the space of the parameters is requiered. The way to analyse the amplification factor is explained further.

B.4 Analysis of the amplification factor

B.4.1 An example : the amplification factor of the Direct Solver

Let us consider the implicit upwind scheme for the Euler equations in which the explicit stage is of third-order while the implicit stage is of first-order. For two-dimensional problem, two reduced wavenumbers have to be defined in each direction : ξ in the x-direction and η in the y-direction. After the Fourier transform, the linear implicit scheme reads :

$$H \cdot \Delta\widehat{w}_{i,j}^{n+1} = -K \cdot \widehat{w}_{i,j}^{n+1} \quad , \tag{B.4.1}$$

with

$$\left\{ \begin{array}{rcl} K & = & \dfrac{2}{3}\left(\dot{D}_1 \cdot z_1 + \dot{D}_2 \cdot z_2\right) + \mathrm{i}\,\left(\dot{A}^E\, s_1\, (1 - \dfrac{1}{3}\, z_1) + \dot{B}^E\, s_2\, (1 - \dfrac{3}{2}\, z_2)\right) \\ H & = & Id \;+\; 2\left(\dot{D}_1 \cdot z_1 \;+\; \dot{D}_2 \cdot z_2\right) \;+\; \mathrm{i}\left(\dot{A}^E s_1 + \dot{B}^E s_2\right) \end{array} \right.$$

Thus the amplification matrix associated with the Direct Solver reads :

$$G_* = Id - H^{-1} \cdot K \tag{B.4.2}$$

The Von Neumann analysis is based on the study of the spectral radius of this amplification

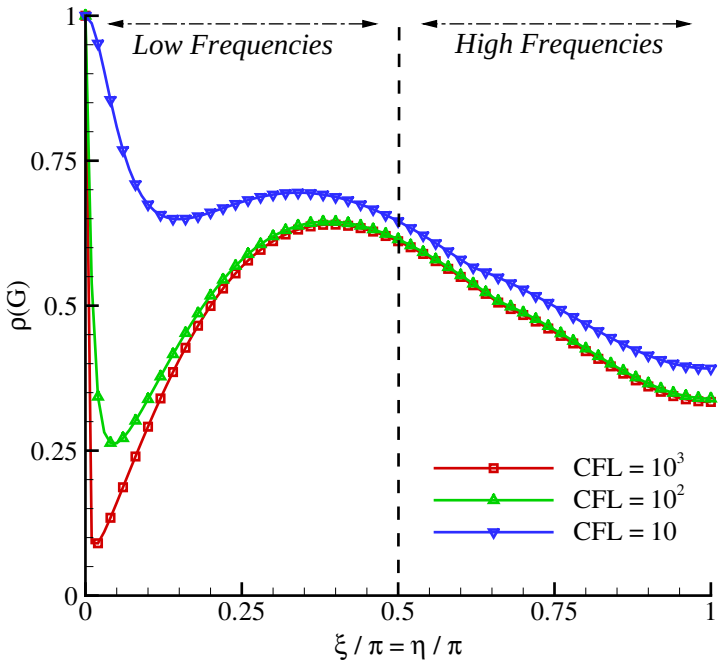


Figure B.4.1: Amplification factor for the Direct Solver, $M = 0.5$, $u/v = 1$.

matrix. If $\rho(G) < 1$ the errors in the system are damped whereas if $\rho(G) > 1$ the errors are amplified leading to instability. The use of a computer program enables us to draw the value of the amplification factor with respect to the wavenumbers ξ and η. For the sake of simplicity, in this appendix, we would consider that $\xi = \eta$. For $M = 0.5$, $u/v = 1$ and several CFL

numbers ($CFL = 10$, $CFL = 10^2$ and $CFL = 10^3$), the amplification factor of the direct solver is represented on figure B.4.1. We can note that, for each CFL number, the amplification factor remains smaller than unity for all wavenumbers which means that the Direct Solver scheme is stable for such conditions. Obviously, the increase of the CFL number enables to improve the efficiency of the scheme. Indeed, we can note that the value of the amplification factor decreases for high CFL numbers. However, we also note that the decrease is not uniform upon the whole space of the wavenumbers. For high frequencies, that is for wavenumbers close to π, increasing the CFL number does not provide a great improvement since the amplification factor does not change a lot. On the other hand, there is a great interest to take high CFL numbers in order to obtain a better damping in the low frequency area (that is for the wavenumbers close to zero).

B.4.2 Link with the convergence rate

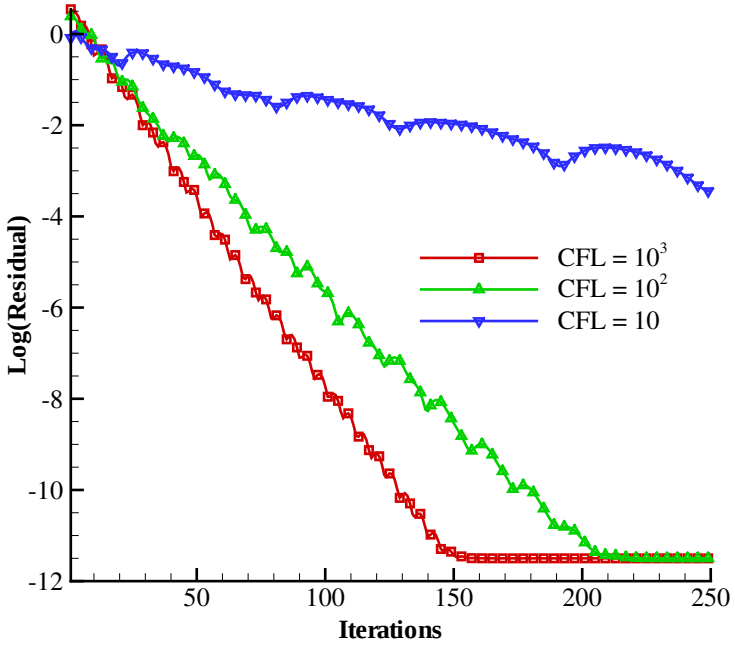


Figure B.4.2: Convergence history in bump-channel with 3600 elements, $M = 0.5$.

In this appendix, we want to show that the results from the Von Neumann analysis are meaningful for practical computations. We computed the flow over a bump in a channel at $M = 0.5$ using a Block Line Jacobi implicit treatment with 10 sub-iterations. The plot of the amplification factor of such a method is very close to the one obtained with the Direct Solver scheme. We used a mesh with 3600 elements and we chose the same CFL numbers than for the Von Neumann analysis. The results are presented in figure B.4.2. As forcasted by the Von Neumann analysis, there is a great difference between the convergence rate obtained with $CFL = 10$ and $CFL = 10^3$. Besides, the convergence rates with $CFL = 10^2$ and $CFL = 10^3$ are very close as expected.
We can conclude from these results that the convergence rate of a method is mainly driven by its efficiency in the low frequency area. Indeed, for high frequencies, the amplification factor is approximatively the same for all CFL numbers while it is very different for low frequencies. The practical computations on an academic test-case of a flow over a bump in a channel outline the

strong influence of the choice of the CFL number on the convergence rate, and this behaviour can be only due to the differences observed in the low frequency domain.

B.4.3 Effect of the mesh size

Let us consider a domain with a length of $2L$ and a constant mesh size δx. We showed in section B.2 that the wavelengths which can be captured on such a domain range from $2\delta x$ to $2L$. The corresponding reduced wavenumbers range from $\pi\delta x/L$ to π. Thus, the size of the spectrum relies on the mesh size. We saw that the region around π is related to the short wavelengths while the region around 0 represents the long wavelengths. Let us precise how the notion of short or long wavelength is mesh dependent.

Let us consider three meshes, a coarse mesh, a medium mesh and a fine mesh defined as :

$$\delta x_c = 2 \cdot \delta x_m = 4 \cdot \delta x_f \tag{B.4.3}$$

The minimum wavenumber ($k = 2\pi/\lambda$) obtained on these three meshes is the same, namely :

$$k_c^{min} = k_m^{min} = k_f^{min} = \frac{\pi}{L} \ . \tag{B.4.4}$$

On the other hand, the maximum wavenumbers are :

$$k_c^{max} = \frac{\pi}{\delta x_c} \ , \quad k_m^{max} = \frac{2 \cdot \pi}{\delta x_c} \ , \quad k_f^{max} = \frac{4 \cdot \pi}{\delta x_c} \ . \tag{B.4.5}$$

Now, let us consider the reduced wavenumber whose definition is given by :

$$\phi = k \cdot \delta x \tag{B.4.6}$$

It appears clearly that the reduced wavenumber π would correspond to a different wavenumber with respect to the mesh size. For the finest mesh, we have :

$$k_c^{max} \cdot \delta x_f = \frac{\pi}{4} \ , \quad k_m^{max} \cdot \delta x_f = \frac{\pi}{2} \ , \quad k_f^{max} \cdot \delta x_f = \pi \ , \tag{B.4.7}$$

while for the coarse mesh, k_m^{max} and k_f^{max} are not represented so we have :

$$k_c^{max} \cdot \delta x_c = \pi \ . \tag{B.4.8}$$

Thus, for the coarse mesh, the wavenumber k_c^{max} belongs to the high frequency area whereas for the fine mesh, it belongs to the low-frequency area. There is a shift towards the low-frequency as the mesh size diminishes. The high frequencies of a coarse mesh moved to the low-frequency part of the spectrum (*cf.* figure B.4.3) when the mesh size falls off. As a result, the same wavelength would be damped differently with respect to the mesh size. For instance, for the Direct Solver scheme using $CFL = 10^3$, the wavelength related to k_c^{max} would be damped according three different amplification factors :

$$\rho\left(G(k_c^{max})\right)|_c = \rho\left(G(\pi)\right) = 0.33 \ ,$$

$$\rho\left(G(k_c^{max})\right)|_m = \rho\left(G(\pi/2)\right) = 0.6 \ ,$$

$$\rho\left(G(k_c^{max})\right)|_f = \rho\left(G(\pi/4)\right) = 0.57 \ .$$

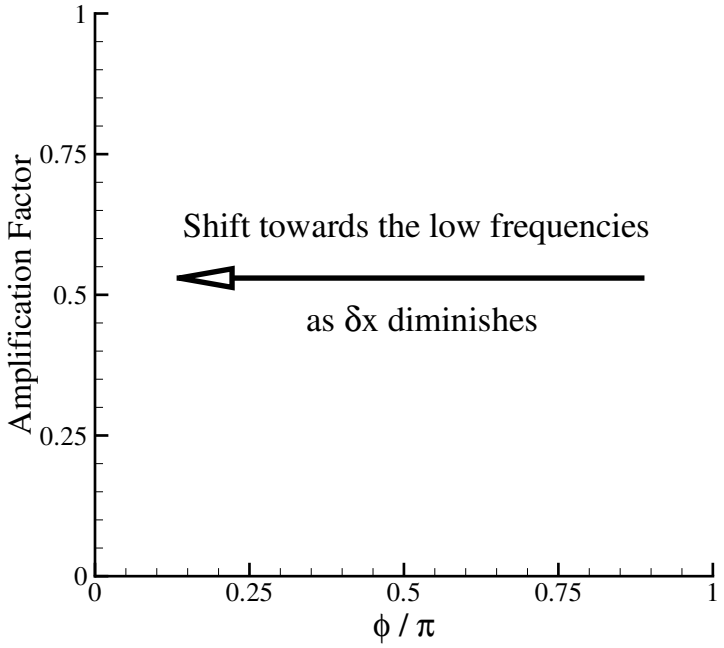


Figure B.4.3: Low-frequency shift

The other effect of the mesh size is related to the accuracy. As the mesh becomes finer, the representation of the longest wavelength is more accurate and the minimum reduced wavenumber becomes very close to zero. The method becomes less efficient because the amplification factor tends to unity for such reduced wavenumbers. Moreover, we showed that the efficiency upon low-frequency area mainly drives the convergence rate, hence, a slowdown of the convergence rate would be expected as the mesh size becomes finer. In practice, this phenomena is truly observed.

Bibliography

[1] J. Amaladas and H. Kamath. Implicit and multigrid procedures for steady-state computations with upwind algorithms. *Computers & Fluids*, 28:187–212, 1999.

[2] T. Barth and D. Jespersen. The design and application of upwind schemes on unstructured meshes. *AIAA Paper*, pages 89–0366, 1989.

[3] R. Beam and R. Warming. On the construction and application of implicit factored schemes for conservation laws. *SIAM-AMS Proceedings*, 11:85–129, 1978.

[4] A. Beccantini, F. Dabbene, S. Kudriakov, J. Magnaud, H. Paillere, and E. Studer. Simulation of hydrogen release and combustion in large scale geometries: Models and methods. In *Int. Conf. on Supercomputing in Nuclear Application*, Paris, France, Sept. 22–24 2003.

[5] A. Beccantini and K. S. Unstructured finite volume schemes for the low mach number compressible Navier-Stokes equations. Internal report of DM2S SFME/LTMF/RT/02-048/A, CEA, Saclay, France, 2002.

[6] A. Bentaïb, J. Vendel, H. Simon, L. Blumenfeld, I. Tkatschenko, and H. Paillère. Air-steam tests in the mistra facility : Experimental results and validation of the lumped-parameter / cfd tonus code. In *ERMSAR 2005, European Review Meeting on Severe Accident Research*, Aix-en-Provence, France, Nov. 14-16 2005.

[7] H. Bijl and P. Wesseling. A unified method for computing incompressible and compressible flows in boundary-fitted coordinates. *Journal of Computational Physics*, 141:153–173, 1998.

[8] W. Briley and H. McDonald. Solution of the multidimensional compressible Navier-Stokes equations by a generalized implicit method. *Journal of Computational Physics*, 24:372–397, 1977.

[9] W. Briley and H. McDonald. An overview and generalization of implicit navier-stokes algorithms and approximate factorization. *Computers & Fluids*, 30:807–828, 2001.

[10] P. Buelow, D. Schwer, J. Feng, and C. Merkle. A preconditioned dual-time, diagonalized adi scheme for unsteady computations. *AIAA Paper*, pages 97–2101, 1997.

[11] P. Buelow, S. Venkateswaran, and C. Merkle. Effect of aspect ratio on convergence. *AIAA Journal*, 32:2401–2408, 1994.

[12] P. Buelow, S. Venkateswaran, and C. Merkle. Stability and convergence analysis of implicit upwind schemes. *Computers & Fluids*, 30:961–988, 2001.

[13] S. Chapuliot, C. Gourdin, T. Payen, J. Magnaud, and A. Monavon. Hydro-thermal-mechanical analysis of thermal fatigue in a mixing tee. *Nuclear Engineering and Design*, 39:617–, 2005.

[14] Y.-H. Choi and C. Merkle. The application of preconditioning in viscous flows. *Journal of Computational Physics*, 105:207–223, 1993.

[15] A. Chorin. A numerical method for solving the incompressible viscous flow problems. *Journal of Computational Physics*, 2:12, 1967.

[16] C. Corre. *Méthode de relaxation par lignes pour des schémas implicites de type Lax-Wendroff en aérodynamique stationnaire.* Phd thesis, Ecole Nationale Supérieure d'Arts et Métiers, Paris, France, 1995.

[17] C. Corre. *Contribution à la Simulation et à l'Analyse des Écoulements Compressibles.* Diplôme d'habilitation à diriger des recherches, Université Pierre et Marie Curie (Paris VI), Paris, France, 2004.

[18] C. Corre, K. Khalfallah, and A. Lerat. Line-relaxation methods for a class of centered schemes. *Computer Fluid Dynamics Journal*, 5:213–246, 1996.

[19] C. Corre, K. Khalfallah, and A. Lerat. Line-relaxation methods for a class of centred schemes. *Computational Fluid Dynamics Journal*, 5:213–246, 1996.

[20] C. Corre and K. Kloczko. Schémas implicites efficaces pour tout régime d'écoulement. In *CANUM2004 36ème Congrès National d'Analyse Numérique*, Obernai, France, 31 mai - 4 juin 2004.

[21] C. Corre and A. Lerat. Efficient calculation of 3-D transonic flows. *Lectures Notes in Physics*, 515, 1998.

[22] C. Corre and A. Lerat. Stability and efficiency of implicit residual-based compact schemes. In M. Hafez and D. Caughey, editors, *Computing the Future IV - Frontiers of CFD - 2004*, 2005.

[23] D. Darmofal and K. Siu. A robust locally preconditionned multigrid algorithm for the Euler equations. *AIAA Paper*, 1998.

[24] D. Drikakis and W. Rider. *High Resolution Methods for Incompressible and Low Speed Flows.* Springer Verlag, 2004.

[25] J. Edwards and M.-S. Liou. Low-diffusion flux-splitting methods for flows at all speeds. *AIAA Journal*, 36:1610–1617, 1998.

[26] S. Gros and T. Kloczko. Implicitation du schéma AUSM+ par une méthode sans matrice - application aux écoulements compressibles stationnaires à faibles nombres de Mach. Internal report of DM2S SFME/LTMF/RT/03-033/A, CEA, Saclay, France, 2003.

[27] H. Guillard and R. Abgrall. *Modélisation numérique des fluides compressibles.* Series in Applied Mathematics, Gauthier-Villars, Paris, 2001.

[28] H. Guillard and C. Viozat. On the behaviour of upwind schemes in the low mach number limit. *Computers & Fluids*, 28(1):63–86, 1999.

[29] F. Harlow and A. Armsden. A numerical fluid dynamics calculation method for all flow speeds. *Journal of Computational Physics*, 8:197, 1971.

[30] G. Hauke and T. Hughes. A comparartive study of different sets of variables for solving compressible and incompressible flows. *Computer Methods Appl. Mech. Eng.*, 153:1–44, 1998.

[31] C. Hirsch. *Numerical Computation of Internal and External Flows, vol. 1: Fundamentals of Numerical Discretization*. Wiley, 1989.

[32] L.-W. Hu, J. Lee, P. Saha, and M. Kazimi. Thermal stripping in lwr piping system. Technical Report MIT-NSP-TR-017, Center for Advanced Nuclear Energy Systems (CANES), MA, USA, 2003.

[33] L.-W. Hu, K. Nagasawa, P. Hejzlar, and M. Kazimi. Thermal striping in lwr piping systems. Technical Report MIT-NSP-TR-007, Center for Advanced Nuclear Energy Systems (CANES), MA, USA, 2001.

[34] A. Jameson. Time-dependent calculations using multigrid, with application to unsteady flow past airfoils and wings. *AIAA Paper*, pages 91–1569, 1991.

[35] A. Jameson and T. Baker. Solution of the euler equations for complex configurations. AIAA Paper.

[36] A. Jameson and D. Caughey. How many steps are required to solve the Euler equations of steady, compressible flow: In search of a fast solution algorithm. *AIAAP*, (2001-2673), 2001.

[37] A. Jameson, W. Schmidt, and E. Turkel. Numerical solutions of the euler equations by finite volume methods using runge-kutta time-stepping schemes. AIAA Paper.

[38] A. Jameson and S. Yoon. Lower-upper implicit schemes with multiple grids for the Euler equations. *AIAA Journal*, 25, 1987.

[39] K. Khalfallah, G. Lacombe, and A. Lerat. Analysis of implicit treatments for a centered Euler solver. *Computers & Fluids*, 22:381–406, 1993.

[40] T. Kloczko. Effets du choix des variables pour la méthode sans matrice et de l'introduction d'une pression de jauge pour les écoulements faiblement compressibles. Internal report of DM2S SFME/LTMF/RT/05-036/A, CEA, Saclay, France, 2005.

[41] T. Kloczko and C. Corre. Méthodes implicites efficaces pour tout régime d'écoulements. Internal report of DM2S SFME/LTMF/RT/04-030/A, CEA, Saclay, France, 2004.

[42] D. Kwak, C. Kiris, and C. Kim. Computational challenges of viscous incompressible flows. *Computers & Fluids*, 34:283–299, 2005.

[43] A. Lerat and C. Corre. A residual-based compact scheme for the compressible Navier-Stokes equations. *Journal of Computational Physics*, 170:642–675, 2001.

[44] A. Lerat and C. Corre. Residual-based compact schemes for the multidimensional hyperbolic systems of conservation laws. *Computers & Fluids*, 31:639–661, 2002.

[45] A. Lerat, J. Sides, and V. Daru. An implicit finite-volume method for solving the euler equations. Lecture Notes in Physics, 17:343–349, 1982.

[46] M.-S. Liou and J. Edwards. AUSM schemes and extension for low Mach and multiphase flows. Technical Report 1999-03, VKI Lecture Series, 1999.

[47] H. Lomax, T. Pulliam, and D. Zingg. *Fundamentals of Computational Fluid Dynamics*. Springer, Berlin.

[48] H. Luo, J. Baum, and R. Löhner. A fast, matrix-free implicit method for compressible flows on unstructured grids. *Journal of Computational Physics*, 146:664–690, 1998.

[49] H. Luo, J. Baum, and R. Löhner. An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grids. *Computers & Fluids*, 30:137–159, 2001.

[50] H. Luo, J. Baum, and R. Löhner. A fast, matrix-free implicit method for computing low-Mach number flows on unstructured grids. *International Journal of Computational Fluid Dynamics*, 14:133–157, 2001.

[51] H. Luo, J. Baum, and R. Löhner. Extension of Harten-Lax-van Leer scheme for flows at all speeds. *AIAA Journal*, 43:1160–1166, 2005.

[52] R. MacCormack. Current status of numerical solutions of the Navier-Stokes equations. *AIAA Paper*, pages 85–0032, 1985.

[53] R. MacCormack. The solution of the Navier-Stokes equations using Gauss-Seidel line relaxation. *Computers & Fluids*, 17:135–150, 1989.

[54] R. MacCormack. Efficient matrix decomposition for implicit algorithms. In *15^{th} Intl. Conf. on Num. Meth. in Fluid Dynamics*, number 490, pages 237–242, Monterey, CA(USA), 1996. Lecture Notes in Physics.

[55] R. MacCormack. A new implicit algorithm for fluid flow. *AIAA Paper*, pages 97–2100, 1997.

[56] R. MacCormack. Iterative modified approximate factorization. *Computers & Fluids*, 30:917–925, 2001.

[57] R. MacCormack and G. Candler. The solution of the navier-stokes equations using gauss-seidel line-relaxation. *Computers & Fluids*, 17(1):135–150, 1989.

[58] B. Müller. Low Mach number asymptotics of the Navier-Stokes equations and numerical implications. Technical Report 1999-03, VKI Lecture Series, 1999.

[59] K. Nerinckx, J. Vierendeels, and E. Dick. Mach-uniformity through the coupled pressure and temperature correction algorithm. *Journal of Computational Physics*, 206:597–623, 2005.

[60] W. Noh. CEL : a time-dependant two-space dimensional coupled Eulerian-Lagrangian Code. *Method in Computational Physics*, pages 117–179, 1964.

[61] H. Paillère, P. L. Quéré, C. Weisman, J. Vierendeels, E. Dick, M. Braack, F. Dabbene, A. Beccantini, E. Studer, T. Kloczko, C. Corre, V. Heuveline, M. Darbandi, and S. Hosseinizadeh. Modelling of natural convection flows with large temperature differences: A benchmark problem problem for low Mach number solvers. part 2. contributions to the june

2004 conference. *ENSAIM: Mathematical Modelling and Numerical Analysis*, 39:617–621, 2005.

[62] S. Pandya, S. Venkateswaran, and T. Pulliam. Implementation of preconditioned dual-time procedures in OVERFLOW. *AIAA Paper*, pages 2003–0072, 2003.

[63] R. Peyret and T. Taylor. Computational methods for fluid flows, 1983.

[64] R. Pulliam, T.H. MacCormack and S. Venkateswaran. Convergence characteristics of several approximate factorization methods. In *ICNMDF98,* 16^{th} *International Conference on Numerical Method in Fluid Dynamics*, number 515, pages 409–414, Arcachon, France, 1998. Lecture Notes in Physics.

[65] T. Pulliam and D. Chaussée. A diagonal form of an implicit approximate-factorisation algorithm. *Journal of Computational Physics*, 39:347–363, 1981.

[66] C. R. *Mécanique expérimetale des Fluides, tome 2: Dynamiques des Fluides réels.* Masson, Paris.

[67] P. L. Roe. Approximate Riemann solvers, parameters vectors and difference schemes. *JCP*, 43:357–372, 1981.

[68] C. Rossow. A blended pressure/density based method for the computation of incompressible and compressible flows. *Journal of Computational Physics*, 185:375–398, 2003.

[69] V. Rusanov. Calculation of interaction of non-steady shock waves with obstacles. *Zhur. Vych. Mat. Fiz.*, 1:267–279, 1961.

[70] J. Sesterhenn, B. Müller, and H. Thomann. On the cancellation problem in calculating compressible low Mach number flows. *Journal of Computational Physics*, 151:597–615, 1999.

[71] D. Sharov and K. Nakahashi. Reordering of 3-d hybrid unstructured grids for vectorized LU-SGS Navier-Stokes computations. *AIAA Paper*, pages 97–0617, 1997.

[72] N. Shende, K. Arora, and N. Balakrishnan. Convergence acceleration using imlicit relaxation procedures for cell centre and cell vertex finite volume schemes. Fluid Mechanics Report 2001FM03, Indian Institute of Science, Bangalore, India, 2001.

[73] P. Sockol. Multigrid solution of the Navier-Stokes equations at low speeds with large temperature variations. *Journal of Computational Physics*, 192:570–592, 2003.

[74] R. Swanson and E. Turkel. On central difference and upwind schemes. *Journal of Computational Physics*, 101:292–306, 1992.

[75] E. Turkel. Preconditioned methods for solving the incompressible and low Mach compressible equations. *Journal of Computational Physics*, 72:277–298, 1987.

[76] E. Turkel. Preconditioning techniques in computational fluid dynamics. *Annu. Rev. Fluid Mech.*, 31:385–416, 1999.

[77] E. Turkel, R. Radespiel, and N. Kroll. Assessment of preconditioning methods for multidimensional aerodynamics. *Computers & Fluids*, 26(6):613–634, 1997.

[78] B. Van Leer. Towards the ultimate conservative difference scheme, v. a second order sequel to godunov's method. *Journal of Computational Physics*, 32:101–136, 1979.

[79] B. Van Leer. Flux vector splitting for the euler equations. In Springer-Verlag, editor, *Proc. 8th International Conference on Numerical Methods in Fluid Dynamics*, Berlin, 1981.

[80] B. Van Leer, W. Lee, and P. Roe. Characteristic time-stepping or local preconditioning of the Euler equations fluid dynamics. *AIAA Paper*, pages 91–1552, 1991.

[81] S. Venkateswaran and C. Merkle. Analysis of preconditioning methods for Euler and Navier-Stokes equation. Technical Report 1999-03, VKI Lecture Series, 1999.

[82] S. Venkateswaran, C. Merkle, X. Zeng, and D. Li. Influence of large scale pressure changes on preconditioned solutions at low speeds. *AIAA Journal*, 42:2490–2498, 2004.

[83] C. Viozat. Implicit upwind schemes for low Mach number compressible flows. Rapport de Recherche 3084, INRIA, 1997.

[84] C. Viozat. *Calcul d'écoulements stationnaires et instationnaires à petits nombres de Mach, et en maillages étirés.* Phd thesis, Université de Nice Sophia-Antipolis, Sophia-Antipolis, France, 1998.

[85] J. Weiss and W. Smith. Preconditioning applied to variable and constant density flows. *AIAA Journal*, 33:2050–2057, 1995.