UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS
FACULTÉ DES SCIENCES

HABILITATION À DIRIGER DES RECHERCHES
EN SCIENCES

MÉTHODES NUMÉRIQUES PERFORMANTES EN MAILLAGES NON-STRUCTURÉS ET APPLICATIONS EN MÉCANIQUE DES FLUIDES COMPRESSIBLES

High-performance numerical methods on unstructured meshes
with applications to compressible fluid dynamics

Présentée par: Stéphane Lanteri

Le 5 Décembre 2003 devant le jury composé de :

Rapporteurs	:	Rémi Abgrall	-	Professeur à l'Université de Bordeaux 1
		David Keyes	-	Professor at Columbia University, NY
		Olivier Pironneau	-	Professeur à l'Université Pierre et Marie Curie
Président	:	Jacques Blum	-	Professeur à l'Université de Nice-Sophia Antipolis
Examinateurs	:	Frédéric Desprez	-	Directeur de Recherche à l'INRIA Rhône-Alpes
		Frédéric Poupaud	-	Professeur à l'Université de Nice-Sophia Antipolis

Contents

Remerciements - Acknowledgments

Les travaux présentés dans ce manuscrit ont été réalisés au sein du projet Sinus de l'INRIA Sophia Antipolis. Je souhaite donc, tout d'abord, remercier avec la plus grande gratitude Alain Dervieux qui, après avoir dirigé mon travail de thèse, m'a apporté sa confiance et m'a soutenu lors de ma candidature comme Chargé de Recherche dans ce même projet. Je voudrais aussi remercier mes collègues chercheurs du projet Sinus, Rémi Abgrall, Jean-Antoine Désidéri et Hervé Guillard pour l'accueil qu'ils m'ont accordé et les collaborations qui nous ont liées de Novembre 1993 à Octobre 2000. Certains des travaux décrits ici n'auraient probablement pas pris forme sans les thèses de Victorita Dolean et Luc Fournier que j'ai eu la chance de co-diriger. Victorita et Luc, je vous remercie encore pour la patience et la persévérance dont vous avez fait preuve tout au long du déroulement de vos thèses. Je souhaite aussi remercier d'autres membres du projet Sinus, post-doctorants ou ingénieurs de recherche, avec lesquels j'ai été amené à collaborer: Angello Iollo, Nathalie Blaszka-Marco, Emmanuel Briand, Gilles Carré et Gilles Carte. Mes remerciements vont par ailleurs à des collaborateurs, chercheurs ou professeurs, externes au projet Sinus mais impliqués dans des études communes, en partie rapportées ici: Herman Deconinck du VKI, Frédéric Nataf du CMAP, Olivier Penanhoat de la SNECMA, Bertand Mantel et Jacques Périaux de Dassault, Patrick Vuillermoz du CNES. Je n'oublie pas Charbel Farhat avec qui j'ai été amené à interagir durant ma thèse à l'occasion de mes premiers pas en calcul parallèle, et qui m'a ensuite accueilli comme post-doctorant au CSSC de l'Université du Colorado à Boulder avant que je ne rejoigne de nouveau l'INRIA où notre collaboration sur le thème de l'interaction fluide/structure s'est poursuivie.

Monsieur Jacques Blum a accepté de présider ce jury, j'en suis très honoré et je l'en remercie.

C'est un grand honneur que d'avoir eu pour rapporteurs Messieurs Rémi Abgrall et Olivier Pironneau. Je les remercie très sincèrement pour le temps précieux qu'ils ont consacré à l'évaluation de mes travaux de recherche.

I would like to express my deep thanks to Professor David Keyes who accepted to act as a referee for my Habilitation. His participation to the jury was for me an immense honour and a great pleasure.

Je remercie sincèrement Monsieur Frédéric Poupaud d'une part, pour avoir été correspondant scientifique à l'Université de Nice-Sophia Antipolis pour cette Habilitation et, d'autre part, pour sa participation au jury.

Mes remerciements vont aussi à Monsieur Frédéric Desprez pour l'intérêt qu'il a porté à mon travail et sa participation au jury.

En Novembre 2000, Serge Piperno m'a accueilli les bras ouverts au sein du projet Caiman. Outre une disponibilité de tous les instants pour me permettre de m'impliquer progressivement dans les activités du projet, il s'est aussi fait un objectif de m'offrir les meilleures conditions de travail pour la préparation de mon Habilitation. Je le remercie chaleureusement pour cela et lui en suis très reconnaissant.

Je souhaite aussi remercier très chaleureusement Loula Fezoui pour son soutien constant et ses encouragements répétés depuis mon arrivée au projet Caiman.

Merci aux autres membres de Caiman, permanents ou non, pour l'amitié dont ils ont fait part à mon égard depuis mon arrivée dans ce projet et notamment à Nathalie Bartoli, Nicolas Canouet, Martine Chane Yook, Said El Kasmi, Gilles Fourestey, Nathalie Glinsky, Christel Luquet, Maud Meriaux et Guillaume Sylvand. Je n'oublie pas Sabine Barrère que je remercie pour son aide de tous les instants dans l'organisation de la soutenance de mon Habilitation.

Un grand merci à l'INRIA Sophia Antipolis, au personnel adiminstratif et du centre de documentation, ainsi qu'aux membres du service SEMIR, pour leur disponibilité et leur compétence.

Enfin, ces dix dernières années de travail à l'INRIA n'auraient certainement pas abouti de la même façon sans la présence à mes côtés et le réconfort permanent de Florence, Marie et Manon, et le soutien de ma famille.

Introduction

Since the early 80s, parallel computers have progressively appeared as the only computer systems capable of handling the challenging applications from various research fields and industrial sectors. Scientific computing offers a large number of such computational applications in physics, mathematics, chemistry and finance among others. This is particularly true for Computational Fluid Dynamics (CFD) which is the domain of application of the research works reported here. After a period of rapid changes in parallel computer architectures that has witnessed supercomputers such as the SIMD (Single Instruction Multiple Data) Connection Machine system or the MIMD (Multiple Instruction Multiple Data) Intel iPSC, Intel Paragon, Cray T3D and SGI Power Challenge Array systems, a more or less steady situation has been reached today, partly due to the emergence of standards in parallel programming models and languages. Roughly speaking, four classes of parallel computing platforms can be exploited nowadays:

- shared memory SMP (Symmetric MultiProcessor) systems such as the SGI Origin series,
- distributed memory clusters based on standard Personal Computer (PC) technology, essentially from Intel and AMD, interconnected by high performance networks (Gigabit Ethernet, Myricom Myrinet, Dolphins SCI, etc.),
- hybrid shared memory/distributed memory systems consisting of clusters of SMPs or more simply, clusters of multiprocessor PCs,
- parallel vector supercomputer systems.

Parallel vector supercomputer systems can rely on a distributed, shared or even a hybrid shared memory/distributed memory organization as it is the case for the Earth Simulator which is currently ranked #1 in the list of TOP 500 supercomputer sites[1] (version of November 2003).

For the numerical solution of systems of Partial Differential Equations (PDEs), the efficient use of such computing platforms requires addressing various topics ranging from computer science concerns to more numerical analysis issues. In this document, we illustrate these two aspects in the context of the calculation of compressible flows that are modeled by the systems of Euler (case of an inviscid flow) and Navier-Stokes (case of a laminar or turbulent flow) equations. Moreover, we consider numerical methods that are designed to work with unstructured finite element type discretizations of the underlying computational domain.

This document is organized as follows. In chapter 1, we first recall the expressions of the systems of non-linear PDEs modeling compressible flows. Then, we describe the main characteristics of the unstructured mesh flow solvers that have been considered as starting points to our contributions. In particular, we present two space approximation methods on tetrahedral meshes that form the basis of these flow solvers: a vertex centered mixed element/volume formulation (MEVF) and a MutliDimensional High Resolution (MDHR) class of compact schemes. As mentioned above, our contributions are concerned both with computer science and numerical analysis aspects of the calculation of compressible flows. In this respect, chapter 2 briefly reviews a widely adopted parallelization strategy for the underlying unstructured mesh flow solvers and discusses in more details its application to the MEVF formulation. In this chapter, we also present some performance results on various clusters of PCs. Chapter 3 is devoted to our contributions on parallel multigrid methods for the acceleration of steady flow calculations for a typical three-dimensional flow calculation. Two multigrid methodologies are considered: a linear multigrid by volume agglomeration strategy in conjunction with the MEVF formulation and a non-linear multi-mesh multigrid approach coupled to the MDHR schemes. Chapter 4 is concerned with the design of domain decomposition methods for the solution of hyperbolic and mixed hyperbolic/parabolic systems of PDEs and their application to the calculation of two-dimensional inviscid and laminar viscous flows. These methods have been implemented in conjunction with the MEVF formulation on unstructured triangular meshes. Moreover, our implementation is characterized by an original approach combining a domain decomposition formulation at the global level and a

[1] http://www.top500.org/

multigrid strategy for the solution of local subdomain problems. In this chapter, both steady and unsteady flows are considered. Finally, chapter 4.3 concludes this document by presenting some possible future works.

All the works presented in this document were, and for some of them are still, developed in collaboration with other researchers and students. Moreover, some other works are not reported here but have been published elsewhere.

The studies undertaken during my doctoral thesis[83] are not discussed in this document. These studies were concerned with several aspects of the resolution of the system of Navier-Stokes equations using the MEVF formulation on unstructured triangular meshes:

- the design of a linearized implicit scheme for the calculation of steady flows, in collaboration with L. Fezoui, B. Larrouturou and C. Olivier[54],
- the design of accurate interpolation schemes in conjunction with the MUSCL method and explicit Runge-Kutta time integration schemes for the calculation of unsteady flows, under the supervision of A. Dervieux,
- the parallelization of the resulting Navier-Stokes solver on the Connection Machine CM-2/CM-200 SIMD system[46], in collaboration with C. Farhat (University of Colorado at Boulder) and L. Fezoui.

The parallelization of 2D and 3D unstructured mesh flow solvers based on the MEVF formulation (chapter 2) has been studied in collaboration with C. Farhat (University of Colorado at Boulder), L. Fezoui (INRIA Sophia Antipolis) and M. Loriot (Simulog). I would also like to mention the work [1] conducted in collaboration with R. Abgrall (University of Bordeaux, formerly at INRIA Sophia Antipolis) and T. Sonar (DLR Gottingen) concerning the design and the parallelization of high-order ENO (Essentially Non-Oscillatory) finite volume schemes on unstructured triangular meshes for the calculation of 2D unsteady compressible flows.

The reduction of computing time of realistic flow calculations through the parallelization of the underlying flow solvers offers the possibility to tackle more complex problems characterized by higher CPU and memory requirements. This is typically the case for optimum shape design applications that bring an additional outer loop on top of the flow solver. Indeed, in [37], we have studied a 3D aerodynamic shape optimization problem using a methodology based on a gradient method where the computation of an adjoint state is done with the help of an automated differentiation tool. This work has been undertaken in collaboration with A. Dervieux J.M. Malé, N. Marco and N. Rostaing-Schmidt (INRIA Sophia Antipolis) and B. Stoufflet (Dassault Aviation). Alternatively, a more recent approach to optimum shape design relies on the use of probabilistic methods such as Genetic Algorithms. (GAs). Moreover, GAs are characterized by a high degree of parallelism. Shortly, the outer loop of a GA requires the evaluation of a fitness function for each individual of a population of potential solutions. Here, an individual is an appropriate binary coding of the set of parameters used to represent a shape through an appropriate technique such as Bezier splines and the evaluation of an individual corresponds to the calculation of the steady flow around this shape. Within each optimization iteration, the evaluation of a fitness function for the members of the current population is a fully concurrent process. Taking into account this fact, we have proposed a two-level parallelization strategy of GAs applied to the optimization of 2D airfoil profiles [38]-[86]. Using the mechanism of process group available in MPI, members of the population are distributed in two groups, Within each group, individuals are evaluated successively using a parallel flow solver such as the one described in chapter 2. At each optimization iteration, two individuals are evaluated simultaneously thanks to the definition of the two group of processes. This work has been conducted in collaboration with J.-A.Désidéri, and N. Marco (INRIA Sophia Antipolis) and with B. Mantel and J. Périaux (Dassault Aviation).

Fluid/structure interaction is another domain that can benefit from high performance computing facilities. The study of the interaction between aerodynamic forces and structural deformation (aeroelasticity) is often performed by considering a weakly coupled approach where existing CFD and CSM simulation software are adapted so that they can exchange appropriate physical quantities defined at the interface between the fluid and structure meshes. In addition an Arbitrary Lagrangian Eulerian (ALE) formulation[48] or a moving mesh finite volume formulation[109] is often adopted for the numerical solution of the fluid equations. In that case, the fluid mesh is deformed at each time step in order to account for the structure displacement. In the most common

implementation of a fluid/structure interaction problem, the CFD and CSM simulation software are executed in a staggered fashion with their own time-stepping loops. Within a time step, aerodynamic forces are sent by the flow solver to the structural mechanics code which computes the resulting deformation. The latter is sent back to the flow solver which updates its mesh accordingly and then advance in time the fluid unknowns. In this context, parallel computing can be exploited at two levels: on one hand, both simulation software can be parallelized using a classical SPMD approach such as the one considered in chapter 2; on the other hand, specific coupling algorithms can be designed so as to introduce a certain degree of concurrency in the evaluation of the aerodynamic forces and structural deformation. In [47], we present our first contributions concerning the numerical simulation of 3D non-linear aeroealstic problems. This work has been conducted in collaboration with C. Farhat, M. Lesoinne and P. Stern (University of Colorado at Boulder), H. Guillard, B. N'Konga and N. Maman (INRIA Sophia Antipolis) and S. Piperno (Cermics, Sophia Antipolis).

For realistic flow calculations, parallelization of the underlying flow solvers is mandatory but often not sufficient. New parallel numerical methods must be designed that are efficient both from the parallel and numerical viewpoints. This is especially true for sparse linear system solution methods that, by the way, constitute a central kernel for most applications requiring the numerical solution of systems of PDEs. In our works, algebraic linear or non-linear system solvers have been considered from three perspectives:

- parallel multigrid methods. The contributions described in chapter 3 are the result of several studies that have been conducted in collaboration with the following former members of the Sinus team at INRIA Sophia Antipolis: E. Briand (research engineer), G. Carré (research engineer), G. Carte (research engineer), A. Dervieux, J.-A. Désidéri, L. Fournier (PhD thesis, 2001) and H. Guillard;

- domain decomposition methods. The works reported in chapter 4 have been (and are still) conducted in collaboration with V. Dolean (assistant professor at the University of Evry, formerly at INRIA Sophia Antipolis, PhD thesis in 2001) and F. Nataf (CMAP, Ecole Polytechnique, Palaiseau).

In the above series of works, the general objective is to efficiently use parallel computing facilities in order to solve larger and more complex problems in a reasonable amount of time, through the development of numerical methods whose parallel and numerical efficiencies are weakly dependent on the number of degrees of freedom. In other situations, parallel computing is combined with appropriate numerical methods for achieving quasi-real time simulations. This is for instance the case with active flow control whose objective is to influence the structure of the flow with the aim of increasing or optimizing aerodynamic performances. One central problem in applications is the prohibitive amount of computational resources induced by the repeated application of the flow solver, in particular in the case of three-dimensional, unsteady (possibly turbulent) flows. A promising technique to circumvent this difficulty is the adoption of low order models as governing equations. Such models should provide a qualitative description of the main features of the flow, for example, in terms of lift and drag versus time, while permitting a very economical numerical solution. One way to devise a low order model is to use Proper Orthogonal Decomposition (POD), by which it is possible to extract from a database of reference simulations a certain number of basis functions onto which the Navier-Stokes equations are projected. The Navier-Stokes equations are thus reduced to a finite-dimensional system of non-linear ODEs. If the dimension of this system is reasonably small, the solution can be found with very limited computational effort. In [73] and [72], we report on our contributions concerning the design of such low order Navier-Stokes models in the context of the mixed element/volume formulation described in section 1.2.1 of chapter 1. These studies have been realized in collaboration with A. Dervieux, J.-A. Désidéri and A. Iollo (postdoc).

Chapter 1

Mathematical models and numerical framework

The works reported in this document deal with several aspects of the numerical treatment of systems of non-linear PDEs that model compressible flows. More precisely, from the numerical analysis point of view, our contributions concern the design of parallel algorithms for the solution of the linear or non-linear algebraic systems resulting from the discretization of the Euler and Navier-Stokes equations. In this chapter, we introduce the general framework that defines the basis of our contributions. We first recall the expressions of the systems of Euler and Navier-Stokes equations in the three-dimensional case. Then, we describe the basic principles of two discretization methods that have been adopted in our studies. These methods share the fact that they rely on the use of triangular (2D case) or tetrahedral (3D case) unstructured meshes. Finally, we present the time integration method that has been used in conjunction with both discretization methods, more precisely, a linearized implicit method for which we consider two variants that are respectively adapted to steady and unsteady flows.

1.1 Mathematical models of compressible flows

Let $\Omega \subset \mathbb{R}^3$ be the computational domain of interest and Γ its boundary. Γ is decomposed as the union of a solid boundary Γ_b and a free-stream boundary Γ_∞: $\Gamma = \Gamma_b \cup \Gamma_\infty$.

1.1.1 System of Euler equations

The conservative form of the system of Euler equations modeling compressible inviscid flows is given by:

$$\frac{\partial W}{\partial t} + \vec{\nabla}.\vec{\mathbb{F}}^c(W) = 0 \quad , \quad W = \left(\rho \ , \ \rho\vec{U} \ , \ E\right)^T \quad , \quad \vec{\nabla} = \left(\frac{\partial}{\partial x} \ , \ \frac{\partial}{\partial y} \ , \ \frac{\partial}{\partial z}\right)^T \tag{1.1}$$

where $W = W(\vec{x}, t)$; $\vec{x}$ and t respectively denote the spatial and temporal variables. In the context of hyperbolic PDEs, $\vec{\mathbb{F}}^c(W) = \left(F_x^c(W) \ , \ F_y^c(W) \ , \ F_z^c(W)\right)^T$ is called the convective flux. For the Euler equations, the convective flux components are given by:

$$F_x^c(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(E+p) \end{pmatrix} \quad , \quad F_y^c(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(E+p) \end{pmatrix} \quad , \quad F_z^c(W) = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(E+p) \end{pmatrix} \tag{1.2}$$

In the above expressions ρ denotes the density, $\vec{U} = (u \ , \ v \ , \ w)^T$ the velocity vector, E the total energy per unit of volume.

Moreover, p is the pressure which is deduced from the other variables using the state equation characterizing a perfect gas :

$$p = (\gamma - 1)(E - \frac{1}{2}\rho \parallel \vec{U} \parallel^2)$$

where γ is the ratio of specific heats ($\gamma = 1.4$ for a diatomic gas).

1.1.2 System of Navier-Stokes equations

The conservative form of the system of Navier-Stokes equations modeling compressible laminar viscous flows is written:

$$\frac{\partial W}{\partial t} + \vec{\nabla}.\vec{\mathbb{F}}^c(W) = \frac{1}{\mathsf{Re}}\vec{\nabla}.\vec{\mathbb{F}}^v(W) \tag{1.3}$$

where $\vec{\mathbb{F}}^c(W) = \left(F_x^c(W)\ ,\ F_y^c(W)\ ,\ F_z^c(W)\right)^T$ is the convective flux defined by eq. (1.2) and where $\vec{\mathbb{F}}^v(W) = \left(F_x^v(W)\ ,\ F_y^v(W)\ ,\ F_z^v(W)\right)^T$ denotes the diffusive flux whose components are given by :

$$F_x^v(W) = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + \dfrac{\gamma}{\mathsf{Pr}}\dfrac{\partial e}{\partial x} \end{pmatrix} \quad , \quad F_y^v(W) = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{yz} + \dfrac{\gamma}{\mathsf{Pr}}\dfrac{\partial e}{\partial y} \end{pmatrix} \tag{1.4}$$

$$F_z^v(W) = \begin{pmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} + \dfrac{\gamma}{\mathsf{Pr}}\dfrac{\partial e}{\partial z} \end{pmatrix}$$

where e is the specific internal energy which is computed from the following state equation :

$$e = \frac{E}{\rho} - \frac{1}{2}(u^2 + v^2) = C_v T = \frac{p}{\rho(\gamma - 1)}$$

where T denotes the temperature. In the above expressions, the components of the Cauchy strain tensor are given by :

$$\begin{aligned}
\tau_{xx} &= \frac{2}{3}\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} - \frac{\partial w}{\partial z}\right) & \tau_{yy} &= \frac{2}{3}\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} - \frac{\partial w}{\partial z}\right) \\
\tau_{zz} &= \frac{2}{3}\left(2\frac{\partial w}{\partial z} - \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) & \tau_{xy} &= \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\
\tau_{xz} &= \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) & \tau_{yz} &= \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)
\end{aligned}$$

Two dimensionless numbers appear in the expressions of the Navier-Stokes equations, the Reynolds number and the Prandtl number (which is only related to physical characteristics of the fluid):

$$\mathrm{Re} = \frac{\rho_0 U_0 L_0}{\mu} \quad \text{and} \quad \mathrm{Pr} = \frac{\mu C_p}{k}$$

where μ and k respectively denote the viscosity and thermal conductivity coefficients and where the subscript 0 is used to identify characteristic quantities of the flow.

1.2 Space discretization methods

This section is devoted to the description of the main characteristics of two space approximation methods on tetrahedral meshes that have been considered in our studies: a vertex centered mixed element/volume formulation (MEVF) (subsection 1.2.1) and a MutliDimensional High Resolution (MDHR) class of compact schemes (subsection 1.2.2).

1.2.1 Mixed element/volume formulation

The flow domain Ω is discretized by a triangulation (2D case) or a tetrahedrization (3D case) $\mathcal{T}_h$ where h is the maximal length of the edges of $\mathcal{T}_h$. A vertex of $\mathcal{T}_h$ is denoted by s_i and the set of neighboring vertices of s_i by $N(s_i)$. We associate to each vertex s_i a control volume (or cell) denoted by C_i which is constructed as the union of local contributions from the set of triangles/tetrahedra sharing s_i (see figure 1.1). The boundary of C_i is denoted by ∂C_i and the normal vector exterior to ∂C_i by $\vec{\eta_i} = (\eta_{ix}, \eta_{iy}, \eta_{iz})$.

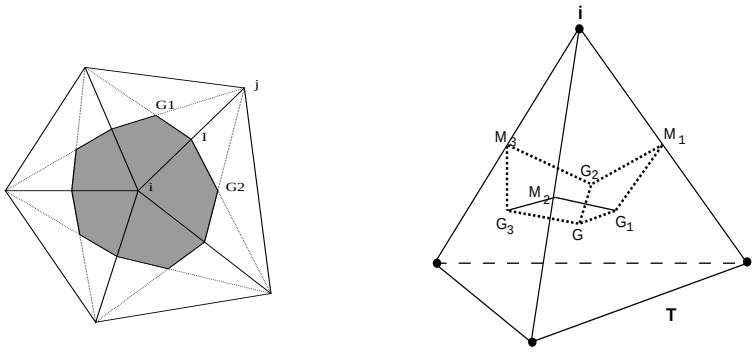


Figure 1.1: The median cell on a triangular mesh (2D case) and the contribution of a tetrahedron to a control volume (3D case)

The space discretization method considered here, from now on referred as a mixed element/volume formulation, combines the following elements:

- a vertex centered finite volume formulation involving upwind schemes for the calculation of the convective fluxes,
- a MUSCL (Monotonic Upstream Schemes for Conservation Laws) technique for the extension to second order accuracy in the calculation of the convective fluxes,
- a finite element formulation (P1 Lagrange element) yielding a centered scheme for the calculation of the diffusive fluxes.

A general variational formulation of eq. (1.3) can be written as:

$$\iiint_{S_i} \left(\frac{\partial W}{\partial t} + \vec{\nabla}.\vec{\mathbb{F}}^c(W) \right) \phi_i d\vec{x} = \frac{1}{\mathsf{Re}} \iiint_{S_i} \vec{\nabla}.\vec{\mathbb{F}}^v(W) \phi_i d\vec{x} \tag{1.5}$$

where S_i is the support of the test function ϕ_i. In the mixed element/volume formulation, the test function is chosen differently according to the flux under consideration. For the convective flux term, the test function is the characteristic function of the control surface C_i:

$$\phi_i = \psi_i \ , \quad S_i = C_i$$

which is such that $\psi_i = 1$ on C_i and 0 elsewhere.

For what concern the diffusive flux term, the test function is the P_1 nodal basis function (linear and continuous on S_i) associated to vertex s_i

$$\phi_i = \varphi_i^\tau \quad \text{and} \quad S_i = \bigcup_{\tau \ , \ s_i \in \tau} \tau \equiv K(s_i) \quad \text{with} \quad \phi_i(s_j) = \delta_{ij} \tag{1.6}$$

Then, the consequences for eq. (1.5) are:

- for the convective flux term:

$$\iiint_{C_i} \vec{\nabla}.\vec{\mathbb{F}}^c(W) \psi_i d\vec{x} = \int_{\partial C_i \bigcup (\partial C_i \cap \Gamma)} \vec{\mathbb{F}}^c(W).\vec{\eta} \psi_i dl - \iiint_{C_i} \vec{\mathbb{F}}^c(W).\nabla \psi_i d\vec{x} \tag{1.7}$$

In eq. (1.7), the second term of the right-hand side is equal to zero since the function ψ_i is constant on C_i.

- for the diffusive flux term:

$$\iiint_{K(s_i)} \frac{1}{\mathsf{Re}} \vec{\nabla}.\vec{\mathbb{F}}^v(W) \varphi_i^\tau d\vec{x} = \frac{1}{\mathsf{Re}} \int_{K(s_i) \cap \Gamma} \vec{\mathbb{F}}^v(W).\vec{\eta} \varphi_i^\tau dl - \frac{1}{\mathsf{Re}} \iiint_{K(s_i)} \vec{\mathbb{F}}^v(W).\nabla \varphi_i^\tau d\vec{x} \tag{1.8}$$

The consistency of the mixed element/volume formulation introduced above has been discussed by Arminjon and Dervieux[7] and rigorously established by Mer[100]. Moreover, Rostand et Stoufflet[123] have studied different formulations for the approximation of the Navier-Stokes equations: a Galerkin finite element formulation[6], a centered mixed finite element/finite volume formulation[90] and the mixed formulation adopted in this study. In particular, the authors have shown that these three formulations are equivalent when applied to the discretization of the system of linearized Euler equations (i.e. when the convective flux terms are linear functions of W).

The discrete equation associated to vertex s_i is obtained by evaluating:

- the convective flux on the boundary $\partial C_i \bigcup (\partial C_i \cap \Gamma)$,
- the diffusive flux on the finite element support of the basis function ϕ_i. In addition, in the evaluation of eq. (1.8), we do not take into account the boundary integrals on $\partial C_i \cap \Gamma_w$ and $\partial C_i \cap \Gamma_\infty$ for reasons that will be discussed in section 1.3.

Then, eq. (1.5) becomes:

$$\iiint_{C_i} \frac{\partial W}{\partial t} d\vec{x} + \int_{\partial C_i \bigcup (\partial C_i \cap \Gamma)} \vec{\mathbb{F}}^c(W).\vec{\eta} dl = -\frac{1}{\mathsf{Re}} \sum_{\tau \in K(s_i)} \iiint_{\tau} \vec{\mathbb{F}}^v(W).\nabla \varphi_i^\tau d\vec{x} \tag{1.9}$$

where φ_i^τ is the P1 basis function associated to element τ.

1.2.1.1 Numerical treatment of the convective flux term

The numerical calculation of the convective flux term relies on a finite volume approximation of the corresponding term of eq. (1.9), This term can be decomposed as:

$$\begin{aligned}\int\limits_{\partial C_i \bigcup(\partial C_i \cap \Gamma)} \vec{\mathbb{F}}^c(W).\vec{\eta} dl &= \sum_{j\in N(s_i)} \int\limits_{\partial C_{ij}} \vec{\mathbb{F}}^c(W).\vec{\eta} dl & < 1 > \\ &+ \int\limits_{\partial C_i \cap \Gamma_w} \vec{\mathbb{F}}^c(W).\vec{\eta} dl + \int\limits_{\partial C_i \cap \Gamma_\infty} \vec{\mathbb{F}}^c(W).\vec{\eta} dl & < 2 >\end{aligned} \tag{1.10}$$

where $\partial C_{ij} = \partial C_i \cap \partial C_j$. In this study, the convective flux terms $< 1 >$ and $< 2 >$ of (1.10) are computed using upwind schemes that are well adapted to the hyperbolic nature of the system of Euler equations. A first order conservative and consistent finite volume approximation of term $< 1 >$ is written:

$$< 1 > = \ W_i^{n+1} - W_i^n + \Delta t \sum_{j\in N(s_i)} \Phi(W_i^n, W_j^n, \vec{\eta}_{ij}) \tag{1.11}$$

where Φ denotes a numerical flux function which is such that:

$$\begin{cases} \Phi(W_i, W_j, \vec{\eta}_{ij}) &= -\Phi(W_i, W_j, -\vec{\eta}_{ij}) \\ \Phi(W_i, W_i, \vec{\eta}_{ij}) &= \vec{\mathbb{F}}^c(W_i).\vec{\eta}_{ij} \end{cases}$$

for any states W_i and W_j and for any vector $\vec{\eta}_{ij}$. The numerical flux function is an approximation of the convective flux through the interface ∂C_{ij} between control volumes C_i and C_j:

$$\Phi_{ij} \equiv \Phi(W_i, W_j, \vec{\eta}_{ij}) \approx \int\limits_{\partial C_{ij}} \vec{\mathbb{F}}^c(W).\vec{\eta} dl \tag{1.12}$$

We have that:

$$\vec{\eta}_{ij} = \int\limits_{\partial C_{ij}} \vec{\eta} d\sigma = \vec{\eta}_1 + \vec{\eta}_2 \tag{1.13}$$

The normal vectors $\vec{\eta}_1$ and $\vec{\eta}_2$ are represented on figure 1.2.

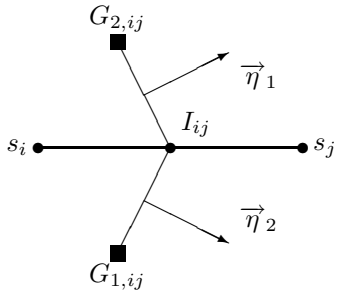


Figure 1.2: Interface ∂C_{ij} between two vertices s_i and s_j

The Jacobian matrix A^c is defined by:

$$A^c(W, \vec{\eta}) = \nabla_W \vec{\mathbb{F}}^c(W).\vec{\eta}$$

where $\nabla_W \vec{\mathbb{F}}^c = \left(\frac{\partial F_x^c}{\partial W} \ , \ \frac{\partial F_y^c}{\partial W} \ , \ \frac{\partial F_z^c}{\partial W} \right)^T$.

From the hyperbolicity of the Euler equations, we know that A^c is diagonalizable with real eigenvalues:

$$\left\{ \begin{array}{lcl} \lambda_1(W,\vec{\eta}) & = & \lambda_2(W,\vec{\eta}) = \lambda_3(W,\vec{\eta}) = \vec{U}\cdot\vec{\eta} \\ \\ \lambda_4(W,\vec{\eta}) & = & \lambda_1(W,\vec{\eta}) + c\|\vec{\eta}\| \\ \\ \lambda_5(W,\vec{\eta}) & = & \lambda_1(W,\vec{\eta}) - c\|\vec{\eta}\| \end{array} \right.$$

where $c = \sqrt{\frac{\gamma p}{\rho}}$ denotes the sound speed. Then, the diagonalization of A^c is written:

$$A^c(W,\vec{\eta}) = T(W,\vec{\eta})\Lambda(W,\vec{\eta})T^{-1}(W,\vec{\eta})$$

On possible strategy to compute the numerical flux Φ_{ij} at the interface between two control volumes is based on the solution of a local one-dimensional (in the direction of the normal vector $\vec{\eta}_{ij}$) Riemann problem defined at the interface ∂C_{ij}:

$$\left\{ \begin{array}{l} \dfrac{\partial W}{\partial t} + \dfrac{\partial}{\partial \eta}\left[\vec{\mathbb{F}}^c(W).\vec{\eta}\right] = 0 \\ \\ W(\vec{X},t) = \left\{ \begin{array}{ll} W_i & \text{if} \quad \vec{X} \in C_i \\ W_j & \text{if} \quad \vec{X} \in C_j \end{array} \right. \end{array} \right. \tag{1.14}$$

where W_i and W_j are two states given at the left and right sides of ∂C_{ij}: In practice, this Riemann problem is often solved approximately using appropriate Riemann solvers. The construction of such schemes is discussed in several textbooks such as the one of Toro[137]. The method adopted in this study has been proposed by Roe[120]. It consists in approximating the solution of the Riemann problem (1.14) through the linearization of the term $\dfrac{\partial}{\partial \eta}\left[\vec{\mathbb{F}}^c(W).\vec{\eta}\right]$ which is then replaced by $\mathcal{A}_{ij}\dfrac{\partial W}{\partial \eta}$ where $\mathcal{A}_{ij}$ is a shorthand notation for $\mathcal{A}_{ij}(W_i, W_j, \vec{\eta}_{ij})$. $\mathcal{A}_{ij}$ is the Jacobian matrix of Roe which is such that it:

1. preserves the hyperbolicity of the original system of PDEs system that is, the diagonalization of $\mathcal{A}_{ij}$ results in real eigenvalues and linearly independent eigenvectors;

2. is consistent with the Jacobian matrix A^c:

$$\mathcal{A}_{ii} \equiv \mathcal{A}_{ii}(W_i, W_i, \vec{\eta}_{ij}) = A^c(W_i, \vec{\eta}_{ij})$$

3. insures a conservation principle through discontinuities:

$$\left(\vec{\mathbb{F}}^c(W_i) - \vec{\mathbb{F}}^c(W_j)\right)\cdot\vec{\eta}_{ij} = \mathcal{A}_{ij}(W_i, W_j, \vec{\eta}_{ij})(W_i - W_j)$$

In practice, the Jacobian matrix that characterizes Roe's scheme is evaluated as:

$$\mathcal{A}_{ij}(\widetilde{W}_{ij}) \equiv A^c(\widetilde{W}_{ij}, \vec{\eta}_{ij}) \tag{1.15}$$

where $\widetilde{W}_{ij}$ is an average state between states W_i and W_j which, in the case of Roe's scheme, is given by:

$$W_i = \begin{pmatrix} \rho_i \\ \rho_i u_i \\ \rho_i v_i \\ \rho_i w_i \\ E_i \end{pmatrix} \quad , \quad W_j = \begin{pmatrix} \rho_j \\ \rho_j u_j \\ \rho_j v_j \\ \rho_j w_j \\ E_j \end{pmatrix} \quad , \quad W_{ij} = \begin{pmatrix} \widetilde{\rho}_{ij} \\ \widetilde{\rho}_{ij}\widetilde{u}_{ij} \\ \widetilde{\rho}_{ij}\widetilde{v}_{ij} \\ \widetilde{\rho}_{ij}\widetilde{w}_{ij} \\ \widetilde{E}_{ij} \end{pmatrix} \tag{1.16}$$

and:

$$\left\{\begin{array}{lcl} \widetilde{\rho}_{ij} & = & \dfrac{\sqrt{\rho_i}\rho_i+\sqrt{\rho_j}\rho_j}{\sqrt{\rho_i}+\sqrt{\rho_j}} \\ \widetilde{u}_{ij} & = & \dfrac{\sqrt{\rho_i}u_i+\sqrt{\rho_j}u_j}{\sqrt{\rho_i}+\sqrt{\rho_j}} \\ \widetilde{H}_{ij} & = & \dfrac{\sqrt{\rho_i}H_i+\sqrt{\rho_j}H_j}{\sqrt{\rho_i}+\sqrt{\rho_j}} \end{array}\right. \quad , \quad \widetilde{v}_{ij}=\frac{\sqrt{\rho_i}v_i+\sqrt{\rho_j}v_j}{\sqrt{\rho_i}+\sqrt{\rho_j}} \quad , \quad \widetilde{w}_{ij}=\frac{\sqrt{\rho_i}w_i+\sqrt{\rho_j}w_j}{\sqrt{\rho_i}+\sqrt{\rho_j}}$$

where H denotes the total enthalpy per unit of volume defined by the relation:

$$H=\frac{E+p}{\rho}=\frac{\gamma p}{(\gamma-1)\rho}+\frac{u^2+v^2+w^2}{2}$$

Then, the numerical flux function Φ_{ij} characterizing Roe's scheme is written:

$$\Phi_{ij}=\Phi(W_i,W_j,\vec{\eta}_{ij})=\frac{\vec{\mathbb{F}}^c(W_i)+\vec{\mathbb{F}}^c(W_j)}{2}.\vec{\eta}_{ij}-d(W_i,W_j,\vec{\eta}_{ij}) \tag{1.17}$$

where $d(W_i,W_j,\vec{\eta}_{ij})$ corresponds to a numerical diffusion term:

$$d(W_i,W_j,\vec{\eta}_{ij})=|\mathcal{A}_{ij}|\frac{W_i-W_j}{2}$$

The numerical flux function (1.17) defines an upwind scheme for the calculation of the convective flux (1.12). Moreover, the third property of the Jacobian matrix $\mathcal{A}_{ij}$ allows to simplify the calculation of the convective flux. Indeed, since:

$$\left\{\begin{array}{lcl} A^{c,+}(W,\vec{\eta}) & = & T(W,\vec{\eta})\Lambda^+(W,\vec{\eta})T^{-1}(W,\vec{\eta}) \\ \Lambda^+(W,\vec{\eta}) & = & \mathsf{diag}(\lambda_k^+(W,\vec{\eta})) \\ \lambda_k^+(W,\vec{\eta}) & = & \max(\lambda_k(W,\vec{\eta}),0) \end{array}\right. \quad \left\{\begin{array}{lcl} A^{c,-}(W,\vec{\eta}) & = & T(W,\vec{\eta})\Lambda^-(W,\vec{\eta})T^{-1}(W,\vec{\eta}) \\ \Lambda^-(W,\vec{\eta}) & = & \mathsf{diag}(\lambda_k^-(W,\vec{\eta})) \\ \lambda_k^-(W,\vec{\eta}) & = & \min(\lambda_k(W,\vec{\eta}),0) \end{array}\right.$$

the numerical flux function simplifies as:

$$\Phi(W_i,W_j,\vec{\eta}_{ij})=\vec{\mathbb{F}}^c(W_j).\vec{\eta}_{ij}-A^{c,+}(\widetilde{W}_{ij},\vec{\eta}_{ij})(W_j-W_i) \tag{1.18}$$

or as:

$$\Phi(W_i,W_j,\vec{\eta}_{ij})=\vec{\mathbb{F}}^c(W_i).\vec{\eta}_{ij}+A^{c,-}(\widetilde{W}_{ij},\vec{\eta}_{ij})(W_j-W_i) \tag{1.19}$$

that yield a simpler and lower cost implementation of the numerical flux function.

The numerical calculation of the convective flux using eq. (1.17) is first order accurate in space. The extension to second order accuracy relies on the MUSCL technique proposed by Van Leer[141] and adapted to triangular meshes by Fezoui and Stoufflet[56]. This technique consists in evaluating interpolated states W_{ij} and W_{ji} ate the boundary ∂C_{ij} through the evaluation of appropriate nodal gradients that are used in a first order Taylor expansion of $W(\vec{x},t)$. Then, these interpolated states are used as arguments to the numerical flux function of eq. (1.19). To be more precise, the interpolated states W_{ij} and W_{ji} are defined as:

$$\bar{W}_{ij}=\bar{W}_i+\frac{1}{2}(\vec{\nabla}\bar{W})_i.s_i\vec{s}_j \quad , \quad \bar{W}_{ji}=\bar{W}_j-\frac{1}{2}(\vec{\nabla}\bar{W})_j.s_i\vec{s}_j \tag{1.20}$$

where $\bar{W} = \left(\rho\ ,\ \vec{U}\ ,\ p\right)^T$ — in other words, the interpolation is done using the physical variables instead of the conservative variables. Then, the interpolated states (1.20) are used as arguments to the numerical flux function (1.17). The nodal gradients $(\vec{\nabla}\tilde{W})_i$ are obtained from a weighted average of the P1 Galerkin (centered) gradients computed on each tetrahedron of the finite element support of s_i:

$$\left(\vec{\nabla}\bar{W}\right)_i = \frac{\iiint\limits_{C_i} \vec{\nabla}\bar{W}|_\tau d\vec{x}}{\iiint\limits_{C_i} d\vec{x}} = \frac{1}{\mathsf{vol}(C_i)} \sum_{\tau \epsilon C_i} \frac{\mathsf{vol}(\tau)}{4} \sum_{s_{i_k} \in T} \bar{W}_{i_k} \vec{\nabla}\varphi_{i_k}^\tau \tag{1.21}$$

where $\varphi_{i_k}^\tau$ is the P1 basis function defined at the vertex s_k and associated with the triangle τ. The construction given by eq. (1.20) and (1.21) results in a half-upwind scheme which is second order accurate but can present spurious oscillations in the solution therefore expressing a loss of monotony. A classical way to cure this problem is to make a compromise between the first order and the second order schemes through the introduction of a slope limitation procedure in the interpolated states (1.20) (see for example [52] for more details).

1.2.1.2 Numerical treatment of the diffusive flux term

The numerical calculation of the viscous flux is based on a finite element approximation of the right-hand side of eq. (1.9). More precisely, by assuming that the primitive variables ρ, $\vec{U}$ are p vary linearly on a triangle τ, we obtain:

$$\iiint\limits_\tau \vec{\mathbb{F}}^v(W).\nabla\varphi_i^\tau d\vec{x} = \mathsf{vol}(\tau)\vec{\mathbb{F}}^v(\tau).\nabla\varphi_i^\tau = \Upsilon_{\tau,i}(\tau) \tag{1.22}$$

where $\vec{\mathbb{F}}^v(\tau)$ is a constant vector on triangle τ whose components are computed using the fact that $\vec{U}(\tau) = \frac{1}{3}\sum_{s_i \in \tau} \vec{U}(s_i)$ (since the components of $\vec{U}$ are assumed to be linear on τ).

1.2.2 Multidimensional high resolution schemes

In this section, we briefly recall the basic principles of multidimensional high resolution schemes using a simple scalar linear advection equation in 2D. We do not discuss aspects related to the extension of these schemes to the systems of PDEs modeling compressible flows i.e. the systems of Euler and Navier-Stokes equations. A detailed overview of multidimensional high resolution schemes and their application to 2D and 3D compressible flows can be found in the book [35].

Let us consider the following scalar linear advection problem in 2D defined on the domain Ω with (piecewise) constant advection speed u_i $(i = 1, 2)$:

$$\frac{\partial\phi}{\partial t} + \sum_{i=1}^{2} u_i \frac{\partial\phi}{\partial x_i} = 0 \tag{1.23}$$

This is an initial and boundary value problem where, for well-posedness, the initial condition:

$$\phi\left(\vec{x}, t=0\right) = \phi_0\left(\vec{x}\right), \quad \forall \vec{x} = (x_1, x_2)^T \in \Omega \tag{1.24}$$

and the boundary condition:

$$\phi\left(\vec{x}, t\right) = \phi_{\Gamma^+}\left(\vec{x}, t\right), \quad \forall \vec{x} \in \Gamma^+,\ \forall t > 0 \tag{1.25}$$

must be given. Here Γ^+ is the inlet boundary of Ω, i.e. the part of Γ where the advection vector $\vec{u} = u_i \vec{1}_{x_i}$, which is the characteristic speed of eq. (1.23), enters the domain. Conversely, on $\Gamma^- = \Gamma \backslash \Gamma^+$, called the outlet boundary, where the characteristic leaves the domain, no boundary conditions have to be specified.

1.2.2.1 Basic principles and properties

Assume that the 2D spatial domain Ω is triangulated with triangles of the type given in figure 1.3, where $1-2-3$ are local vertex numbers. The inward scaled normals, i.e. normals with the length of the corresponding edge, are defined for a counter clockwise numbering of the vertices by:

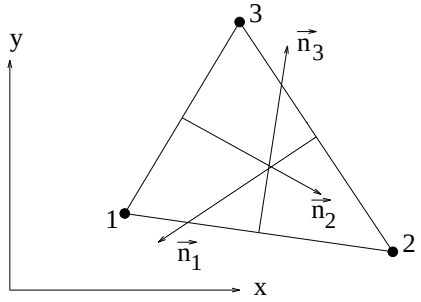


Figure 1.3: A generic triangle with inward scaled normals

$$\vec{n}_1 = (y_2 - y_3)\,\vec{1}_x + (x_3 - x_2)\,\vec{1}_y \tag{1.26}$$
$$\vec{n}_2 = (y_3 - y_1)\,\vec{1}_x + (x_1 - x_3)\,\vec{1}_y \tag{1.27}$$
$$\vec{n}_3 = (y_1 - y_2)\,\vec{1}_x + (x_2 - x_1)\,\vec{1}_y \tag{1.28}$$

where, for convenience, x and y are used to indicate the independent variables. In case the vertex numbering is clockwise the subscripts in the definitions (1.26) to (1.28) must be reversed. It is obvious that:

$$\vec{n}_1 + \vec{n}_2 + \vec{n}_3 = \vec{0} \tag{1.29}$$

In the finite element method the unknown ϕ of eq. (1.23) is assumed to have the following form:

$$\phi\,(\mathbf{x}, t) = \sum_{l=1}^{\#\text{vertices}} N_l\,(\vec{x})\,\phi_l\,(t) \tag{1.30}$$

where ϕ_l is the value of ϕ in vertex s_l and N_l the nodal basis functions that must fulfill the property $N_l(\vec{x}_k) = \delta_{kl}$ (see figure 1.4). We recall that the nodal basis function N_l is obtained by assembling the local P1 basis function φ_l^τ (see eq. (1.6) of subsection 1.2.1) defined on each triangle τ of the finite element support $K(s_l)$ of vertex s_l.

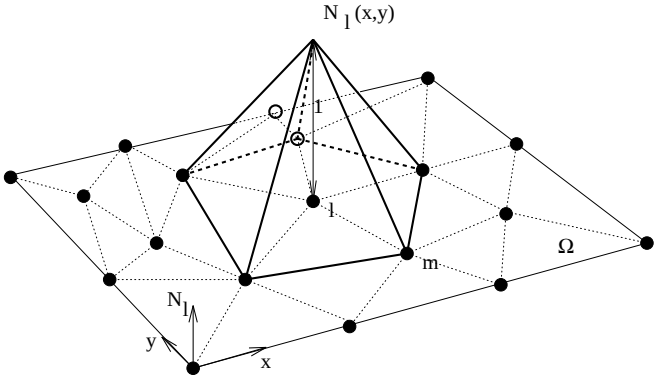


Figure 1.4: Two-dimensional piecewise linear interpolation function for vertex s_l

Then, the Petrov-Galerkin discretization at vertex s_l of eq. (1.23) is written:

$$\iint_{\Omega} w_l(\vec{x}) \sum_{k=1}^{\#\text{vertices}} N_k(\vec{x}) \frac{\partial \phi_k}{\partial t} d\Omega + \iint_{\Omega} w_l(\vec{x}) \sum_{i=1}^{2} u_i \sum_{k=1}^{\#\text{vertices}} \frac{\partial N_k(\vec{x})}{\partial x_i} \phi_k d\Omega = 0 \tag{1.31}$$

Here w_l is the Petrov-Galerkin weight function at vertex s_l, which should obey certain conditions, see [78]. In practice this means that w_l and its first order derivatives must be square integrable. Consequently w_l is bounded. Note that if the weight functions w_l are identical to the basis functions N_l, the classical Galerkin finite element method is obtained. Otherwise the method is called a Petrov-Galerkin finite element method. Usually the global system (1.31) is built as a summation over the individual triangles τ:

$$\sum_{\tau} \left\{ \iint_{\tau} w_l(\vec{x}) \sum_{k=1}^{3} N_k(\vec{x}) \frac{\partial \phi_k}{\partial t} d\tau + \iint_{\tau} w_l(\vec{x}) \sum_{i=1}^{2} u_i \sum_{k=1}^{3} \frac{\partial N_k(\vec{x})}{\partial x_i} \phi_k d\tau \right\} = 0 \tag{1.32}$$

The first integral leads to the product of the mass matrix M, whose element M_{lk} is defined as:

$$M_{lk} = \sum_{\tau} \iint_{\tau} w_l(\vec{x}) N_k(\vec{x}) d\tau \tag{1.33}$$

by the time derivative of ϕ_k. For time dependent problems, the mass matrix must be taken into account to obtain an accuracy in time higher than first order. Note that the mass matrix (1.33) results in a relation between vertices s_l and s_k and consequently a linear system must be solved every time step, even if an explicit time integration method is used. However, in the present work, only steady problems are considered and therefore the w_l in the mass matrix can be chosen differently from the w_l in the spatial part of eq. (1.32). It was found that the most stable formulation was obtained if M was approximated by the lumped Galerkin, $w_l = N_l$, mass matrix :

$$M_{lk}^{Gal} = \sum_{\tau} \iint_{\tau} N_l(\vec{x}) N_k(\vec{x}) d\tau \approx \sum_{\tau \in K(s_l)} \frac{\text{area}(\tau)}{3} \delta_{lk} = S_l \delta_{lk} \tag{1.34}$$

Here δ_{lk} the Kronecker delta function and S_l the area of the median dual cell of vertex s_l, see figure 1.5. $\sum_{\tau \in K(s_l)}$ indicates all triangles which belong to the finite element support (or neighborhood) of vertex s_l (see figure 1.6).

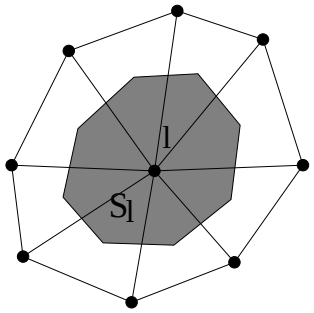


Figure 1.5: Vertex s_l, its immediate neighbors and its median dual cell S_l (shaded region)

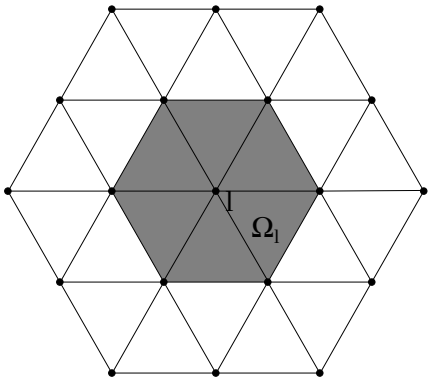


Figure 1.6: The neighborhood $K(s_l)$, shaded region, of vertex s_l

The approximation (1.34) does not change the steady-state solution and, because it is a diagonal matrix, it avoids the solution of a linear system for explicit time integration schemes.

For a linear triangle it can easily be verified that the following relation holds inside triangle T:

$$\frac{\partial N_k(\vec{x})}{\partial x_i} = \frac{n_{k_i}}{2\mathsf{area}(\tau)} \tag{1.35}$$

where n_{k_i} is the component of $\vec{n}_k$ in the x_i direction, i.e. $n_{k_i} = \vec{n}_k \cdot \vec{1}_{x_i}$. Consequently the second integral of equation (1.32) for triangle τ, from now on called R_l^τ, simplifies to:

$$R_l^\tau = \left(\frac{1}{\mathsf{area}(\tau)} \iint\limits_\tau w_l(\vec{x}) d\tau \right) R_\tau \tag{1.36}$$

Here R_τ is the cell residual:

$$R_\tau = \frac{1}{2} \sum_{p=1}^{3} (u_1 n_{p_1} + u_2 n_{p_2}) \phi_p = \sum_{p=1}^{3} k_p \phi_p \tag{1.37}$$

where the upwind parameters k_p are defined as:

$$k_p = \frac{1}{2}(u_1 n_{p_1} + u_2 n_{p_2}) \tag{1.38}$$

Note that in eq. (1.37) use has been made of the assumption that u_i is (piecewise) constant. The form (1.36) can be simplified even further by introducing the distribution coefficient β_l^T for triangle T:

$$\beta_l^T = \frac{1}{S_T} \iint_T w_l(\mathbf{x}) dT, \tag{1.39}$$

leading to:

$$R_l^T = \beta_l^T R_T \tag{1.40}$$

The β_l^T are called distribution coefficients, because they distribute parts of the cell residual R_T to the 3 vertices of triangle T. For consistency the local nodal residuals R_l^T should sum up to R_T, which is equivalent to:

$$\beta_1^T + \beta_2^T + \beta_3^T = 1 \tag{1.41}$$

Combining eq. (1.32), (1.34) and (1.40) results in:

$$\frac{d\phi_l}{dt} + \frac{1}{S_l} \sum_T \beta_l^T R_T = 0 \tag{1.42}$$

The conclusion of this exercise is that any Petrov-Galerkin finite-element scheme for eq. (1.23) with piecewise constant advection speed u_i on linear triangles can be written in the residual distribution form (1.40). The condition that the weight functions w_l are bounded has as a consequence that the distribution coefficients β_l^T are bounded. This implies that every Petrov-Galerkin finite-element scheme is linearity preserving (see discussion below).

The properties of the different schemes depend on the definition of the β_l^T (or equivalently w_l). The only restriction that they must obey is given in eq. (1.41), and consequently degrees of freedom remain to adapt the scheme such that it has the desired properties. The following properties will be discussed in detail in the next sections:

- Multidimensional Upwind ($\mathcal{MU}$) : the multidimensional upwind scheme minimizes the amount of cross-wind diffusion within the class of upwind schemes and consequently gives the most accurate results.
- Positivity ($\mathcal{P}$) : the positivity property guarantees that existing discontinuities are captured monotonically (if the initial solution does not contain overshoots).

- Linearity Preservation ($\mathcal{LP}$) : within the class of schemes considered, i.e. assuming a linear variation of the solution per element, at most linear varying solutions can be reproduced exactly by the numerical scheme.
- Continuity ($\mathcal{C}$) : continuity of the schemes is required to obtain a smooth convergence to the steady-state solution.

The optimal numerical scheme, again within the class considered, for advection problems with discontinuities should have all four properties.

1.2.2.1.1 Multidimensional Upwind ($\mathcal{MU}$). The discretization technique is a finite-element method on an unstructured triangular grid. A condition must be found which guarantees multidimensional upwinding for this kind of schemes. For scalar advection, one can distinguish two types of triangles, one-inflow and two-inflow triangles, of which examples are shown in figure 1.7.

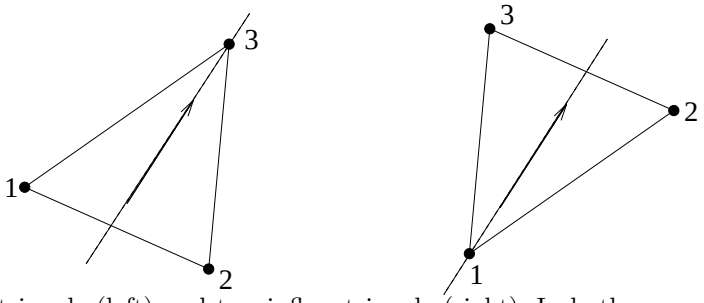


Figure 1.7: One-inflow triangle (left) and two-inflow triangle (right), In both cases the arrow indicates the direction of the streamline

For one-inflow triangles only one of the dot products of the advection vector with the inward scaled normals, the upwind parameters k_p (1.38), is positive; for two-inflow triangles two of the k_p's are positive. As eq. (1.29) holds, and thus:

$$k_1 + k_2 + k_3 = 0 \tag{1.43}$$

It is obvious that these are the only configurations possible. A scheme is now said to be multidimensional upwind if :

$$\mathcal{MU} \ : \ \beta_l^T = 0 \quad \text{if } k_l \leq 0 \tag{1.44}$$

i.e. nothing is distributed to inflow vertices.

1.2.2.1.2 Positivity ($\mathcal{P}$). In a compressible flow, discontinuities like shocks and slip lines can be present. It is desirable that the numerical method captures these phenomena monotonically, i.e. without under- and overshoots. One way to achieve this is to require the scheme to be positive. The semi-discrete version of eq. (1.23) at vertex s_l, in which only the spatial part has been discretized, can be written as:

$$\frac{d\phi_l}{dt} + \sum_m c_{lm} \left(\phi_l - \phi_m\right) = 0 \tag{1.45}$$

Note that for the schemes considered here, only the immediate neighbors have non-zero coefficients c_{lm}. A scheme is said to be positive if:

$$\mathcal{P} \ : \ c_{lm} \geq 0 \quad \forall l, m, \ l \neq m \tag{1.46}$$

which is identical to Jameson's LED (Local Extremum Diminishing) criterion [75]. This ensures that no new extrema are created. It is clear from eq. (1.45) and (1.46) that ϕ_l cannot increase/decrease for a local maximum/minimum. Clearly the positivity property does not guarantee existing overshoots to vanish. Therefore it is possible that there are several (numerical) steady-state solutions depending on the initial conditions, especially for nonlinear schemes and/or problems.

From the positivity concept it is possible to derive a time step restriction. For a general time integration scheme this becomes:

$$\Delta t_l \leq \frac{\mathsf{CFL}}{\sum_{m \neq l} c_{lm}} \tag{1.47}$$

where CFL is a typical constant for the time integration method used.

The positivity condition (1.46) is grid dependent and consequently difficult to impose. Therefore the local positivity property is introduced. This requires that condition (1.46) is obeyed for each individual element. Obviously this is more restrictive, but it has the advantage of being grid independent and thus easy to impose.

1.2.2.1.3 Linearity Preservation ($\mathcal{LP}$). Linearity preservation is the ability of the numerical scheme to reproduce steady linear solutions of eq. (1.23) exactly. Within the class of schemes considered, with linear variation of the solution over the elements, this is the highest possible order of polynomials, which can be calculated exactly. The condition imposed on the residual distribution schemes (1.40) can easily be determined. Consider vertex s_l and its immediate neighbors (see figure 1.5). The numerical steady-state solution is given by, see eq. (1.42):

$$\frac{d\phi_l}{dt} = -\frac{1}{S_l} \sum_T \beta_l^T R_T = 0 \tag{1.48}$$

where S_l is the area of the median dual cell of vertex s_l, see figure 1.5, which enters the formulation due to the mass matrix approximation (1.34). One of the cells T of figure 1.5 is depicted in more detail in figure 1.8. Consequently a sufficient condition for linearity preservation is:

$$\mathcal{LP} \ : \ \beta_l^T \text{ is bounded} \tag{1.49}$$

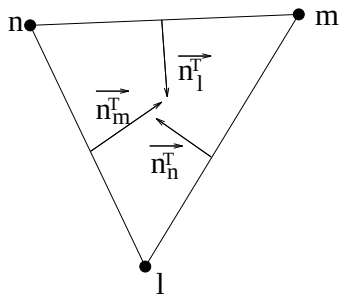


Figure 1.8: Cell T which shares vertex l and its inward scaled normals

Struijs[133] proved that linear schemes cannot have both the positivity and linearity preservation property, which is an equivalent formulation of the famous Godunov theorem[65]. A scheme is called linear if R_l^T, the part of the residual R_T of cell T which is sent to vertex s_l, is a linear function of the unknowns ϕ_m, i.e. the coefficients c_{lm} in eq. (1.45) are independent of ϕ_m.

1.2.2.1.4 Continuity ($\mathcal{C}$). Convergence towards steady-state solutions by means of iterative methods are hampered if the distributions to the vertices discontinuously change from one iteration to the other. Usually a cyclic behavior, known as a limit cycle, is then observed and therefore it is desirable that the schemes are continuous. Two kinds of continuity are distinguished:

- continuity with respect to the solution ϕ_m,
- continuity with respect to the advection vector u_i (only important for non-constant advection speeds).

1.2.2.2 Two numerical schemes

1.2.2.2.1 The N-scheme ($\mathcal{MU}, \mathcal{P}, \mathcal{C}$). The N-scheme is the residual distribution formulation of the first-order multidimensional upwind method. For the triangle shown in figure 1.3, the distributions to the vertices are defined by:

$$R_l^{\mathrm{N}} = k_l^+ \left(\phi_l - \phi_{in}\right) \tag{1.50}$$

where:

$$\phi_{in} = \frac{1}{\sum_{m=1}^{3} k_m^-} \sum_{m=1}^{3} k_m^- \phi_m \tag{1.51}$$

is the (linearly interpolated) state at the inflow point of the triangle, point 1 in figure 1.7 for a two inflow triangle, see also figure 1.7. The parameters k_l^+ and k_l^- are defined as $\max(0, k_l)$ and $\min(0, k_l)$ respectively, in which k_l is the upwind parameter, see eq. (1.38). The multidimensional upwind property is easily checked, as only vertices with $k_l > 0$ receive a contribution due to the multiplication with k_l^+.

If eq. (1.51) is substituted in (1.50) the following alternative formulation of the N-scheme is obtained:

$$R_l^{\mathrm{N}} = \sum_{m=1}^{3} c_{lm} \left(\phi_l - \phi_m\right), \quad c_{lm} = \frac{k_l^+ k_m^-}{\sum_{p=1}^{3} k_p^-} \tag{1.52}$$

This form is identical to the one used in the positivity definition, eq. (1.46). From the formulation (1.52) it is obvious the coefficients $c_{lm} \geq 0$ and thus the N-scheme is positive. Clearly, the scheme is linear as well.

The distribution coefficients β_l^{N} are not bounded. This means that the N-scheme is only first order accurate, which is in agreement with Godunov's theorem. Due to this non-boundedness of β_l^{N}, and consequently of the weight function w_l, see eq. (1.39) , the N-scheme does not belong to the class of Petrov-Galerkin finite-element schemes. The operations $k_l^+ = \max(0, k_l)$ and $k_l^- = \min(0, k_l)$ are continuous at $k_l = 0$ and therefore the continuity of the scheme is ensured.

1.2.2.2.2 The PSI or limited N-scheme ($\mathcal{MU}, \mathcal{LP}, \mathcal{P}, \mathcal{C}$). The PSI (Positive Stream-wise Invariant) scheme [134] was the first scheme, which had all the properties defined at the beginning of this section. The original formulation of the PSI-scheme is given in [134]:

$$\beta_l^{\mathrm{PSI}} = \frac{\max\left(0, \beta_l^{\mathrm{N}}\right)}{\sum_{m=1}^{3} \max\left(0, \beta_m^{\mathrm{N}}\right)} \tag{1.53}$$

where:

$$\beta_l^{\mathrm{N}} = \frac{R_l^{\mathrm{N}}}{R_T} \tag{1.54}$$

Later Sidilkover and Roe[127] showed that a linearity-preserving, positive scheme could be constructed from any first-order, positive scheme by applying a symmetric limiter function (with some additional constraints) to the distributions of the first-order scheme.

If the first-order scheme is the N-scheme, the limiter must be applied only if two inflow parameters k_l are positive, since in the one-target case the N-scheme contributions are both positive and linearity preserving. Again assume that k_2 and k_3 are the positive inflow parameters and the corresponding N-scheme distributions are given by R_2^{N} and R_3^{N} respectively. Then the limited N-scheme is defined as :

$$R_2^{\mathrm{LN}} = R_2^{\mathrm{N}} \left[1 - \psi\left(\frac{-R_3^{\mathrm{N}}}{R_2^{\mathrm{N}}}\right)\right] \tag{1.55}$$

$$R_3^{\mathrm{LN}} = R_3^{\mathrm{N}} \left[1 - \psi\left(\frac{-R_2^{\mathrm{N}}}{R_3^{\mathrm{N}}}\right)\right] \tag{1.56}$$

where $\psi(r)$ is a limiter function with the following properties :

- consistency : $\psi(1) = 1$
- symmetry : $\psi\left(\frac{1}{r}\right) = \frac{\psi(r)}{r}$

The consistency property ensures linearity preservation. Note that the argument r of the limiter ψ equals 1 if $R_2^{\mathrm{N}} = -R_3^{\mathrm{N}}$ which is the case if the cell residual $R_T = 0$. If the limiters are applied, see eq. (1.55) and (1.56), it is clear that R_2^{LN} and R_3^{LN} are zero only if $\psi(1) = 1$. The symmetry property ensures conservation, i.e. $R_2^{\mathrm{LN}} + R_3^{\mathrm{LN}} = R_T$.

From eq. (1.55) and (1.56) it can be seen that the N-scheme distribution is multiplied by a factor. To ensure that the limited scheme is locally positive, this factor must be positive. In combination with the consistency and symmetry condition, this gives an additional condition for the limiter function:

$$0 \leq \psi(r) \leq 1 \tag{1.57}$$

Sidilkover and Roe[127] showed that this condition can be relaxed somewhat for global positivity, for example for linear advection with a constant advection speed on a uniformly triangulated structured grid, the global positivity condition becomes:

$$0 \leq \psi(r), \frac{\psi(r)}{r} \leq 2 \tag{1.58}$$

However in general this is not the case and in practice condition (1.57) is used. If the so-called minmod limiter is chosen:

$$\psi(r) = \max[0, \min(r, 1)] \tag{1.59}$$

the expressions (1.55) and (1.56) can be simplified, if use is made of the typical property of the minmod limiter:

$$\psi(r) + \psi(1 - r) = 1 \tag{1.60}$$

The distribution to vertex 2 then becomes:

$$R_2^{\mathrm{LN}} \quad = \psi\left(\frac{R_2^{\mathrm{N}}}{R_T}\right) R_T = \psi\left(\beta_2^{\mathrm{N}}\right) R_T \tag{1.61}$$

A similar analysis for R_3^{LN} gives:

$$R_3^{\mathrm{LN}} = \psi\left(\frac{R_3^{\mathrm{N}}}{R_T}\right) R_T = \psi\left(\beta_3^{\mathrm{N}}\right) R_T \tag{1.62}$$

This form of the limited N-scheme shows that a linearity preserving, positive scheme can be obtained by simply limiting the distribution coefficients of the N-scheme. Moreover, if the expressions (1.61) and (1.62) are compared with the original formulation of the PSI-scheme, eq. (1.53), one can see that both schemes are identical. However, the formulation with a limiter function is more general, because other limiters than minmod can be chosen. Since both the limiter and its arguments are continuous functions, the PSI-scheme, or limited N-scheme, is continuous.

1.2.2.3 System schemes

Even though the general concepts of the distribution schemes are easily introduced on a scalar linear advection equation, the ultimate goal is to design robust multidimensional upwind discretization techniques for non-commuting hyperbolic systems in general and the compressible Euler equations in particular. Consider the hyperbolic system in time:

$$\frac{\partial W}{\partial t} + A_i \frac{\partial W}{\partial x_i} = 0 \tag{1.63}$$

Here W is a vector with N elements and the $N \times N$ Jacobian matrices A_i do not commute. As system (1.63) is hyperbolic in time, the generalized upwind parameters K_l:

$$K_l = \frac{1}{d} A_i n_{l_i}, \tag{1.64}$$

where d is the number of spatial dimensions, have a complete set of real eigenvalues and eigenvectors. Consequently K_l can be written as:

$$K_l = R_l \Lambda_l L_l, \tag{1.65}$$

where the columns of R_l contain the right eigenvectors, Λ_l is a diagonal matrix of the eigenvalues and $L_l = R_l^{-1}$. The matrices K_l^+ and K_l^-, generalizing the scalar coefficients k_l^+ and k_l^-, are defined as:

$$K_l^+ = R_l \Lambda_l^+ L_l, \quad K_l^- = R_l \Lambda_l^- L_l \tag{1.66}$$

Here Λ_l^+ contains the positive and Λ_l^- the negative eigenvalues: $\Lambda_l^{\pm} = \frac{1}{2}(\Lambda_l \pm |\Lambda_l|)$. As for the scalar case, the cell residual R_T, which is a vector now, is obtained by integrating the spatial part of equation (1.63) over the control volume T, a triangle or a tetrahedron, resulting in:

$$R_T = \iint_T A_i \frac{\partial W}{\partial x_i} dT = \int_{\partial T} A_i n_i^{\mathrm{ext}}\, W d\partial T \tag{1.67}$$

Assuming A_i constant per cell, and a linear variation of the elements of W (or equivalently integrating the contour integral of eq. (1.67) with the trapezium rule) gives:

$$R_T = \sum_{l=1}^{d+1} K_l W_l \tag{1.68}$$

Exactly as for the scalar case, fractions of R_T are distributed to the vertices and the nodal updates are obtained by assembling the contributions of all cells, leading to the semi-discretization:

$$\frac{dW_l}{dt} = -\frac{1}{S_l} \sum_T \beta_l^T R_T \tag{1.69}$$

Also for the system case the mass matrix approximation (1.34) is used. The coefficients β_l^T are now matrices and for consistency $\sum_m \beta_m^T = \mathrm{Id}$ (although this is not entirely true), where the summation extends over the $d+1$ vertices of the cell.

The extension of scalar numerical schemes presented in subsection 1.2.2.2 to system schemes is described in details in [111] (see also the book [35]). We simply note that the application of the system schemes to the Euler equations needs a quasi-linear form of the equations. To retain conservation and hence capturing of discontinuities with their correct jump relations, a particular quasi-linear form is used, which is simultaneously conservative (see [111]).

1.2.2.4 Hyperbolic/elliptic splitting using preconditioning

Roe *et al.*[121]-[101] have observed that the preconditioning technique of van Leer *et al.*[142], originally developed for speeding up the convergence, also results in the most decoupled form of the Euler equations and that space discretizations based on this form are superior to discretizations of the original equations. We recall below the main characteristics of this preconditioning technique in the two dimensional case.

Consider the set of variables:

$$\partial Q = \begin{pmatrix} \dfrac{\partial p}{\rho c} \\ \partial u \\ \partial v \\ \dfrac{\partial p}{p} - \gamma \dfrac{\partial \rho}{\rho} \end{pmatrix} \tag{1.70}$$

The last component of Q is the differential form of the unscaled entropy $s = \ln \dfrac{p}{\rho^\gamma}$. For this set of variables, the Euler equations take a relatively simple form:

$$\frac{\partial Q}{\partial t} + A_{Q,x} \frac{\partial Q}{\partial x} + A_{Q,y} \frac{\partial Q}{\partial y} = 0 \tag{1.71}$$

with:

$$A_{Q,x} = \begin{pmatrix} u & c & 0 & 0 \\ c & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u \end{pmatrix} \quad \text{and} \quad A_{Q,y} = \begin{pmatrix} v & 0 & c & 0 \\ 0 & v & 0 & 0 \\ c & 0 & v & 0 \\ 0 & 0 & 0 & v \end{pmatrix}$$

Note that the last equation is fully decoupled form the others, ahowing that the entropy is advected along the streamline. An even simpler form is obtained in the streamline coordinate system (ξ, η):

$$\frac{\partial \tilde{Q}}{\partial t} + A_{\tilde{Q},x} \frac{\partial \tilde{Q}}{\partial \xi} + A_{\tilde{Q},y} \frac{\partial \tilde{Q}}{\partial \eta} = 0 \tag{1.72}$$

with:

$$A_{\tilde{Q},x} = c \begin{pmatrix} M & 1 & 0 & 0 \\ 1 & M & 0 & 0 \\ 0 & 0 & M & 0 \\ 0 & 0 & 0 & M \end{pmatrix} \quad \text{and} \quad A_{\tilde{Q},y} = c \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where:

$$\partial\tilde{Q} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \partial Q = \begin{pmatrix} \dfrac{\partial p}{\rho c} \\ \partial\tilde{u} \\ \partial\tilde{v} \\ \dfrac{\partial p}{p} - \gamma\dfrac{\partial \rho}{\rho} \end{pmatrix}$$

is the set of variables in the (ξ, η) system and θ is the flow direction with respect to the cartesian coordinate system. The idea of preconditioning is to change the transient behavior of (1.72) while keeping the same steady-state solution. To accomplish this, the spatial part of (1.72) is multiplied by a matrix P which results in:

$$\frac{\partial\tilde{Q}}{\partial t} + P\left(A_{\tilde{Q},x}\frac{\partial\tilde{Q}}{\partial\xi} + A_{\tilde{Q},y}\frac{\partial\tilde{Q}}{\partial\eta}\right) = 0 \tag{1.73}$$

In order to insure that the character of the original system is conserved, the sign of the eigenvalues of the preconditioned Euler equations must be identical to the sign of the eigenvalues of the original system. Therefore, a necessary condition is that the preconditioning matrix is positive definite. To do so, van Leer *et al.*[142] have proposed the following formulation with the goal of clustering the eigenvalues of the preconditioned system:

$$P = \begin{pmatrix} aM^2 & -aM & 0 & 0 \\ -aM & a+1 & 0 & 0 \\ 0 & 0 & \chi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.74}$$

with:

$$\begin{cases} a &= \dfrac{\chi}{\beta^2} \\ \beta &= \sqrt{\max(\varepsilon^2, |M^2-1|)} \\ \chi &= \dfrac{\beta}{\max(M,1)} \end{cases}$$

where ε is a small value, typically 0.05, to avoid the sonic point singularity. Introducing the "steady-state" characteristic variables:

$$\partial\tilde{W} = \begin{pmatrix} \beta & 0 & M & 0 \\ \beta & 0 & -M & 0 \\ 1 & M & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \partial\tilde{Q} = \begin{pmatrix} \beta\dfrac{\partial p}{\rho c} + M\partial\tilde{v} \\ \beta\dfrac{\partial p}{\rho c} - M\partial\tilde{v} \\ \dfrac{\partial p}{\rho c} + M\partial\tilde{u} \\ \dfrac{\partial p}{p} - \gamma\dfrac{\partial\rho}{\rho} \end{pmatrix} \tag{1.75}$$

transforms the system (1.73) into:

$$\frac{\partial\tilde{W}}{\partial t} + A_{\tilde{W},x}\frac{\partial\tilde{W}}{\partial\xi} + A_{\tilde{W},y}\frac{\partial\tilde{W}}{\partial\eta} = 0 \tag{1.76}$$

with:

$$A_{\tilde{W},x} = \tilde{u}\begin{pmatrix} \chi\nu^+ & \chi\nu^- & 0 & 0 \\ \chi\nu^- & \chi\nu^+ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad A_{\tilde{W},y} = \tilde{u}\begin{pmatrix} \dfrac{\chi}{\beta} & 0 & 0 & 0 \\ 0 & -\dfrac{\chi}{\beta} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where ν^+ and ν^- are functions of the Mach number:

$$\nu^+ \frac{M^2 - 1 + \beta^2}{2\beta^2} \quad \text{and} \quad \nu^- \frac{M^2 - 1 - \beta^2}{2\beta^2}$$

which are bounded between [0,1] and [-1,0] respectively. Clearly, the third and fourth equations are decoupled and their quantities advected along the streamline. These correspond to entropy (the fourth variable) and total enthalpy.

Indeed, it is straightforward to show that the following relation holds:

$$\partial H = c\partial\tilde{W}_3 + \frac{c^2}{\gamma(\gamma-1)}\partial\tilde{W}_4$$

where $\partial\tilde{W}_3$ and $\partial\tilde{W}_4$ are the third and fourth components of $\partial\tilde{W}$. Furthermore, for supersonic flows, $M > 1$, $\nu^+ = 1$ and $\nu^- = 0$, also the first and second equations of system (1.76) are decoupled, and the Euler equations reduce to a set of scalar advection equations. The advection directions of the so-called acoustic variables $\partial\tilde{W}_1$ and $\partial\tilde{W}_2$ are the Mach lines (see fig. 1.9) where the Mach angle μ is the angle between the streamline and the Mach lines:

$$\mu = \arctan\left(\frac{1}{\beta}\right)$$

For subsonic flow, $M < 1$, $\nu^+ = 0$ and $\nu^- = -1$, the acoustic variables form a Cauchy-Riemann type subsystem, which explains the partially elliptic nature of the steady, subsonic Euler equations. As at the characteristic level the entropy and total enthalpy equations are always decoupled, it is possible to use a combination of different schemes, for example the PSI-scheme for the advection of entropy and total enthalpy along the streamline and another scheme (even a classical Lax-Wendroff scheme) for the acoustic subsystem.

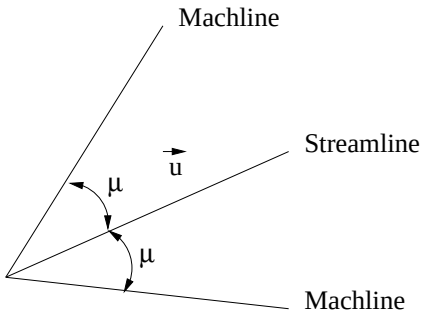


Figure 1.9: Streamline and Mach lines for a two-dimensional supersonic flow

In practice, despite the fact that the characteristic form (1.76) is used to define the upwind parameters while implementing the approximation schemes discussed previsouly, itis stressed that the method is conservative and the conservative varaibles are updated at every time step. This is accomplished by the introduction of consistent nodal values of the characteristic variables:

$$\tilde{W}^{\text{consistent}} = \frac{\partial\tilde{W}}{\partial Z}(\bar{Z})Z$$

for the discretization of system (1.76). In the above equation, Z denotes the so-called Roe parameter vector[120]:

$$Z = \sqrt{\rho}\begin{pmatrix} 1 \\ u \\ v \\ H \end{pmatrix}$$

while the linearization vector $\bar{Z}$ is defined for each tetrahedron by:

$$\bar{Z} = \frac{1}{4}\sum_{k=1}^{4} Z_k$$

The udpate of the conservative variables is then obtained by inverting the steps (1.71), (1.72), (1.73) and (1.76) in reversed order. These steps correspond to the transformation of the conservative variables W to the Q

ones, to the transformation to the streamline frame, to the multiplication with the preconditioning matrix P and to the transformation of the $\tilde{Q}$ variables into the characteristic variables $\tilde{W}$.

We finally note that in three dimensions the same procedure is followed and the preconditioning matrix of van Leer *et al.*[142], in the streamline coordinate system (ξ, η, ζ) and for the set of variables:

$$\partial\tilde{Q} = \begin{pmatrix} \dfrac{\partial p}{\rho c} \\ \partial\tilde{u} \\ \partial\tilde{v} \\ \partial\tilde{w} \\ \dfrac{\partial p}{p} - \gamma\dfrac{\partial\rho}{\rho} \end{pmatrix}$$

is given by:

$$P = \begin{pmatrix} aM^2 & -aM & 0 & 0 & 0 \\ -aM & a+1 & 0 & 0 & 0 \\ 0 & 0 & \chi & 0 & 0 \\ 0 & 0 & 0 & \chi & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

all parameters being unchanged. Then, the three-dimensional version of system (1.76) shows that, as in two space dimensions, total enthalpy and entropy always decouple and are advected along the streamline. However, in contrast to the two-dimensional case, the acoustic equations never decouple and form a 3×3 subsystem, even for supersonic flows.

1.2.2.5 Time integration

The spatial discretization leads to a system of ordinary differential equations in time :

$$\frac{dW}{dt} - R(W) = 0 \tag{1.77}$$

where $W = (W_1, W_2, ..., W_l, ...)^T$ is the vector of nodal states and $R(W)$ the discretized spatial part. Note that for systems, the nodal states W_l are themselves vectors, with each N elements. Both explicit and implicit methods are considered. Eq. (1.77) can be solved using an explicit Runge-Kutta method or a linearized implicit scheme such as the ones discussed in section 1.4 for the mixed element/volume formulation.

1.3 Boundary conditions

1.3.1 Free-Stream conditions

When one is dealing with the numerical simulation of external flows, the computational mesh is generally constructed such that the free-stream boundary Γ_∞, which is an artificial boundary of the computational domain Ω, is located far enough from the boundary Γ_w delimiting the body. In these conditions, the viscous effects are neglected on Γ_∞. Consequently, the corresponding boundary integral term in eq. (1.8) is not taken into account and the flow in the free-stream region is assumed to be uniform:

$$\rho_\infty = 1 \ , \ \vec{U}_\infty = (u_\infty \ , \ v_\infty \ , \ w_\infty)^T \quad \text{with} \quad \| \vec{U}_\infty \| = 1 \ , \ p_\infty = \frac{1}{\gamma M_\infty^2} \tag{1.78}$$

In practice, an *upwind-downwind* flux decomposition is used to compute the corresponding boundary integral term in eq. (1.7). More precisely, this boundary term is evaluated using a non-reflexive version of the Steger and Warming flux decomposition[131]:

$$\int_{\partial C_i \cap \Gamma_\infty} \vec{\mathbb{F}}^c(W).\vec{\eta} dl = A^{c,+}(W_i, \vec{\eta}_{i\infty}).W_i + A^{c,-}(W_i, \vec{\eta}_{i\infty}).W_\infty \tag{1.79}$$

1.3.2 Wall conditions

1.3.2.1 Case of an inviscid fluid

The slipping condition $\vec{U}.\vec{\eta} = 0$ is applied on the boundary Γ_w. This condition is introduced in the corresponding boundary integral term in eq. (1.7). This means that the pressure integral term given by:

$$\int_{\partial C_i \cap \Gamma_w} \vec{\mathbb{F}}^c(W).\vec{\eta} dl = p_i \begin{pmatrix} 0 \\ \eta_{ix} \\ \eta_{iy} \\ \eta_{iz} \\ 0 \end{pmatrix} \tag{1.80}$$

is taken into account in the computation of the flux balance associated to vertex s_i.

1.3.2.2 Case of a viscous fluid

In that case, a no-slip condition needs to be applied on the boundary Γ_w:

$$\vec{U} = 0$$

together with an isothermal wall condition:

$$T = T_w$$

where the wall temperature T_w is computed as:

$$T_w = T_\infty \left(1 + \frac{\gamma - 1}{2} M_\infty^2\right)$$

where M_∞ denotes the free-stream Mach number. The above conditions are applied in a strong way meaning that for a vertex s_i located on Γ_w, $U(s_i)$ is enforced to 0, $T(s_i)$ is set to T_w while the density ρ_i is kept unchanged since the corresponding PDE is hyperbolic (in other words, the mass flux through $\partial C_i \cap \Gamma_w$ is supposed zero). Finally, the total energy per unit of volume and the pressure are updated according to:

$$E_i = \rho_i C_v T_w \quad \text{and} \quad p_i = (\gamma - 1) E_i$$

1.4 Time integration methods

Here we introduce two linearized implicit schemes for the time integration of the semi-discrete equations resulting from the space discretization of the Euler and Navier-Stokes equations using the mixed element/volume formulation described in subsection 1.2.1. Similar schemes in conjunction with the multidimensional high resolution schemes presented in subsection 1.2.2 are discussed in details in [74].

1.4.1 Calculation of steady flows

When one is interested in the calculation of steady flows, time accuracy is not a constraint and the main objective is to design a time integration strategy that allows for a fast convergence to the steady state. Implicit time integration schemes have the necessary damping properties that make them the good candidates for time integration of the semi-discrete equations in this context. Moreover, further acceleration of the convergence is obtained by using

local time stepping strategies. Assuming $W(\vec{x},t)$ is constant on each cell C_i, a linearized backward Euler implicit time integration scheme (see also Fezoui and Stoufflet[56]) is written:

$$\mathsf{vol}(C_i)\left[\frac{dW}{dt}\right]_i^{n+1} + \Psi(W^{n+1})_i = 0 \quad \text{with} \quad \left[\frac{dW}{dt}\right]_i^{n+1} = \frac{\delta W_i^{n+1}}{\Delta t} \tag{1.81}$$

where $W_i^n = W(\vec{x}_i, t^n)$, $t^n = n\Delta t$, $\delta W_i^{n+1} = W_i^{n+1} - W_i^n$ and:

$$\Psi(W)_i = \sum_{j\in N(s_i)} \Phi(W_{ij}, W_{ji}, \vec{\eta}_{ij}) + \int_{\partial C_i \cap \Gamma} \vec{\mathbb{F}}^c(W).\vec{\eta} dl + \frac{1}{\mathsf{Re}} \sum_{\tau \in K(s_i)} \Upsilon_{\tau,i} \tag{1.82}$$

Applying a first order linearization to the nodal flux $\Psi(W^{n+1})_i$ yields the Newton-like formulation:

$$\left(\frac{\mathsf{vol}(C_i)}{\Delta t} + \frac{\partial \Psi(W^n)_i}{\partial W}\right)\delta W_i^{n+1} = -\Psi(W^n)_i \quad , \quad i = 1, \cdots, N_v \tag{1.83}$$

meaning that, at each time step, the following linear system must be solved:

$$P(W^n)\delta W^{n+1} = \left(\mathsf{diag}\left(\frac{\mathsf{vol}(C_i)}{\Delta t}\right) + J(W^n)\right)\delta W^{n+1} = -\Psi(W^n) \tag{1.84}$$

The calculation of the convective and diffusive contributions to the Jacobian matrix $J(W^n)$ is discussed in the following paragraphs.

1.4.1.1 Linearization of the convective flux term

The contribution of the convective flux term to the Jacobian matrix $P(W^n)$ of eq. (1.84) is based on an approximate linearization of the first order numerical flux (1.17) (more precisely, (1.18) or (1.19)). For an edge $[s_i, s_j]$, an implicit version of the numerical flux (1.17) is formally written as:

$$\Phi_{ij}^{n+1} = \Phi(W_i^n, W_j^n, W_i^{n+1}, W_j^{n+1}, \vec{\eta}_{ij})$$

Introducing $U = W_i^n$, $V = W_j^n$, $W = W_i^{n+1}$ and $Z = W_j^{n+1}$ and using a first order Taylor expansion of the implicit flux we obtain:

$$\Phi(U,V,W,Z,\vec{\eta}_{ij}) = \Phi(U,V,\vec{\eta}_{ij}) + \left(\frac{\partial \Phi}{\partial U}\right)(W-U) + \left(\frac{\partial \Phi}{\partial V}\right)(Z-V) \tag{1.85}$$

where:

$$\Phi(U,V,\vec{\eta}_{ij}) \equiv \Phi(W_i^n, W_j^n, \vec{\eta}_{ij})$$

is the explicit flux. Eq. (1.84) can be simplified in the case where the numerical flux function has the form:

$$\Phi(U,V,\vec{\eta}_{ij}) = H_1(U,V,\vec{\eta}_{ij})U + H_2(U,V,\vec{\eta}_{ij})V$$

For example, for the numerical flux function characterizing the approximate Riemann solver of Roe[120] (using eq. (1.19)) we have:

$$\begin{aligned} \Phi(W_i, W_j, \vec{\eta}_{ij}) &= \vec{\mathbb{F}}^c(W_i).\vec{\eta}_{ij} + A^{c,-}(W_i, W_j, \vec{\eta}_{ij})(W_j - W_i) \\ &= A^c(W_i, \vec{\eta}_{ij})W_i + A^{c,-}(\widetilde{W}_{ij}, \vec{\eta}_{ij})(W_j - W_i) \end{aligned}$$

therefore:

$$\left\{ \begin{array}{lcl} H_1(U,V,\vec{\eta}_{ij}) &=& A^c(U,\vec{\eta}_{ij}) - A^{c,-}(U,V,\vec{\eta}_{ij}) \\ H_2(U,V,\vec{\eta}_{ij}) &=& A^{c,-}(U,V,\vec{\eta}_{ij}) \end{array} \right. \tag{1.86}$$

Now, an approximate linearization is obtained if we assume that:

$$\frac{\partial \Phi}{\partial U} \approx H_1(U,V,\vec{\eta}_{ij}) \text{ and } \frac{\partial \Phi}{\partial V} \approx H_2(U,V,\vec{\eta}_{ij}) \tag{1.87}$$

We note that for the numerical flux function characterizing the Steger and Warming scheme[131] (see eq. (1.79)) we obtain:

$$\left\{ \begin{array}{lcl} H_1(U,V,\vec{\eta}_{i\infty}) & = & A^{c,+}(U,\vec{\eta}_{i\infty}) \\ H_2(U,V,\vec{\eta}_{i\infty}) & = & A^{c,-}(U,\vec{\eta}_{i\infty}) \end{array} \right.$$

1.4.1.2 Linearization of the diffusive flux term

Similarly to the previous paragraph, an implicit version of the diffusive flux term (1.22), expressing the contribution of triangle τ[1] to the global flux balance associated to vertex s_i, is written as:

$$\Upsilon_{\tau,i}^{n+1} = \Upsilon_{\tau,i}\left(W_{k1}^n, W_{k2}^n, W_{k3}^n, W_{k1}^{n+1}, W_{k2}^{n+1}, W_{k3}^{n+1}\right)$$

where s_{k1}, s_{k2} et s_{k3} are the vertices of triangle τ (one of them denoting vertex s_i). Proceeding similarly to the linearization of the convective flux we obtain:

$$\begin{array}{rcl} \Upsilon_{\tau,i}^{n+1} & = & \Upsilon_{\tau,i}\left(W_{k1}^n, W_{k2}^n, W_{k3}^n\right) \\ & + & \left(\frac{\partial \Upsilon_{\tau,i}}{\partial W_{k1}^n}\right)\left(W_{k1}^{n+1} - W_{k1}^n\right)\left(\frac{\partial \Upsilon_{\tau,i}}{\partial W_{k2}^n}\right)\left(W_{k2}^{n+1} - W_{k2}^n\right) + \left(\frac{\partial \Upsilon_{\tau,i}}{\partial W_{k3}^n}\right)\left(W_{k3}^{n+1} - W_{k3}^n\right) \end{array} \tag{1.88}$$

In the above expressions, $\Upsilon_{\tau,i}\left(W_{k1}^n, W_{k2}^n, W_{k3}^n\right)$ is given by:

$$\begin{array}{rcl} \Upsilon_{\tau,i}\left(W_{k1}^n, W_{k2}^n, W_{k3}^n\right) & = & \text{vol}\left(\tau\right)\Bigg(F_x^v\left(W_{k1}^n, W_{k2}^n, W_{k3}^n\right)\frac{\partial \phi_i}{\partial x}\mid_\tau + \\ & & \qquad F_y^v\left(W_{k1}^n, W_{k2}^n, W_{k3}^n\right)\frac{\partial \phi_i}{\partial y}\mid_\tau\Bigg) \end{array} \tag{1.89}$$

The expressions of the various terms of eq. (1.88) are obtained thanks to an exact differentiation approach (see [54] for more details).

1.4.2 Calculation of unsteady flows

The time integration method described in the previous section relies on a backward Euler implicit scheme that we rewrite here as:

$$D\frac{W_h^{n+1} - W_h^n}{\Delta t} + \frac{\partial \Psi_h^p(W_h^n)}{\partial W_h}\left(W_h^{n+1} - W_h^n\right) = -\Psi_h^p(W_h^n) \tag{1.90}$$

where D is the diagonal matrix defined by:

$$D_i = \text{vol}(C_i)$$

and where we assume that:

- W_h^n denotes the discrete solution computed at the vertices of the mesh $\mathcal{T}_h$, at time $t^n = t^0 + n\Delta t$;
- $\Psi_h^p(W_h^n)$ denotes the p-order accurate nodal flux whose expression for $p = 2$ is given by eq. (1.82).

[1] Here, we consider the 2D case to simplify the presentation.

In the case $p = 2$, it is a hard task, if not an impossible one, to compute analytically the Jacobian matrix $\frac{\partial \Psi_h^2(W_h^n)}{\partial W_h}$. This matrix is characterized by a sparse irregular symmetric structure but it is a non-symmetric matrix from the non-zero entries point of view. On the other hand, the second order approximation of the numerical fluxes based on the MUSCL technique (see Fezoui and Stoufflet[56] for more details) results in a notable increase of the matrix bandwidth (recall that the actual number of neighbors varies from one vertex to another since the mesh is assumed to be fully unstructured) and may require a considerable amount of memory for its storage even if an appropriate compressed sparse matrix storage format is adopted. When facing such a situation, the strategy generally adopted (as it is the case in the present study) consists in replacing the exact computation of the Jacobian matrix by a an approximate one following a certain number of simplifications. The most notable of these simplifications amounts to base the calculation of the Jacobian matrix on the linearization of a first order accurate convective flux. In these conditions, we obtain the following hybrid scheme:

$$D\frac{W_h^{n+1} - W_h^n}{\Delta t} + \frac{\partial \Psi_h^1(W_h^n)}{\partial W_h}\left(W_h^{n+1} - W_h^n\right) = -\Psi_h^2(W_h^n) \tag{1.91}$$

To be more precise, in the present case, the nodal flux is the sum of a convective term and a diffusive term. In practice, the approximate Jacobian matrix $\frac{\partial \Psi_h^1(W_h^n)}{\partial W_h}$ is computed from:

- an approximate linearization of the (analytical) first order convective flux (see eq. (1.84)-(1.85)),
- an exact linearization of the (discrete) second order diffusive flux (see eq. (1.89)).

It is clear that for steady state computations, eq. (1.91) yields a second order accurate solution of the steady Navier-Stokes equations. eq. (1.91) defines an inexact Newton method which is not characterized by a quadratic convergence as $\Delta t \longrightarrow +\infty$. However, the resulting linear system is easier to solve (in particular, the corresponding matrix has a minimal bandwidth). It is possible to show that for certain linear model problems, the Jacobian matrix verifies a diagonal dominance property allowing the use of simple relaxations methods (Jacobi or Gauss-Seidel) to approximately solve the linear systems. Then, the idea is to construct a second order accurate implicit scheme relying on the approximate Jacobian matrix $\frac{\partial \Psi_h^1(W_h^n)}{\partial W_h}$. Such a time integration scheme has been proposed by Martin and Guillard[89] and we recall this approach here.

If we assume that the time derivative term of eq. (1.81) is now discretized using:

$$\frac{dW}{dt} \approx \frac{aW(t+\Delta t) - V^\star\left[W(t), W(t-\Delta t), \cdots, W(t-j\Delta t)\right]}{\Delta t} \tag{1.92}$$

then the new scheme is based on a general defect-correction type iteration:

$$\left\{\begin{array}{rcl} \mathcal{W}_h^0 & = & \mathcal{W}^0\left[W_h^n, W_h^{n-1}, \cdots, W_h^{n-j}\right] \\ \mathcal{W}_h^{s+1} & = & \mathcal{W}_h^s - \Delta t\left[a\mathsf{Id} + \Delta t\frac{\partial \Psi_h^m(W_h^n)}{\partial W_h}\right]^{-1} \times \\ & & \left(\frac{a\mathcal{W}_h^s - V^\star\left[W_h^n, W_h^{n-1}, \cdots, W_h^{n-j}\right]}{\Delta t} + \Psi_h^p(\mathcal{W}_h^s)\right) \end{array}\right. \tag{1.93}$$

where:

- $V^\star\left[W(t), W(t-\Delta t), \cdots, W(t-j\Delta t)\right]$ denotes a linear function of its j arguments,
- each iteration of the form (1.93) defines an operator $G_h^s(\Delta t)$ that associates to $V^\star$ the vector $\mathcal{W}_h^s$.

The mathematical properties of the time integration scheme based on eq. (1.93) are analyzed in details in Martin and Guillard[89]. One result which is relevant to our study says that if Ψ_h^2 denotes a second-order approximation of the nodal flux then:

- if $\mathcal{W}_h^0 = 0$, $s_p \geq 2$ iterations of eq. (1.93) are necessary to obtain a second order accurate (in space and time) approximation of $W(\vec{x}, t)$,
- if $\mathcal{W}_h^0 = W_h^n$, $s_p \geq 1$ iterations of eq. (1.93) are sufficient to obtain a second order accurate (in space and time) approximation of $W(\vec{x}, t)$.

In summary, the second order linearized implicit scheme used in practice consists in the following two steps:

$$\begin{aligned}
\left[\frac{3}{2}\mathsf{Id} + \Delta t \frac{\partial \Psi_h^1(W_h^n)}{\partial W_h}\right](U - W_h^n) &= -\left(\frac{3}{2}W_h^n + \Delta t \Psi_h^2(W_h^n)\right) + \left(2W_h^n - \frac{1}{2}W_h^{n-1}\right) \\
\left[\frac{3}{2}\mathsf{Id} + \Delta t \frac{\partial \Psi_h^1(W_h^n)}{\partial W_h}\right](W_h^{n+1} - U) &= -\left(\frac{3}{2}U + \Delta t \Psi_h^2(U)\right) + \left(2W_h^n - \frac{1}{2}W_h^{n-1}\right)
\end{aligned} \tag{1.94}$$

Chapter 2

Parallelization strategies on unstructured meshes

2.1 Preamble

In the past ten years, the parallelization of CFD solvers relying on finite volume or finite element type formulations on unstructured meshes has been the subject of active studies worldwide as witnessed by the numerous contributions in the proceedings of the main conference in the field[1]. In most of the cases, the parallelization of these solvers relies on a classical and widely used SPMD approach that combines a partitioning of the underlying mesh and a message-passing programming model. Such a strategy is discussed in [84] in the context of the parallelization of a CFD solver for the Euler and Navier-Stokes equations that implements the mixed element/volume formulation on unstructured tetrahedral meshes described in section 1.2.1 of chapter 1. However, one particularity of our study lies in the partitioning strategy and its influence on the parallel performances. Indeed, for the mixed finite element/volume formulation under consideration, two partitioning strategies can be considered: an *overlapping* or vertex oriented strategy and a *non-overlapping* or element oriented strategy. Both strategies can be characterized a priori in terms of computational and communication complexities. In this chapter, we briefly review the characteristics of these two approaches and we present updated performance results.

2.2 Computational and parallel implementation issues

2.2.1 Identification of the main computational kernels

From the description of the mixed element/volume numerical scheme in section 1.2.1 of chapter 1, it appears that the resulting flow solver contains essentially two types of elementary computations: on one hand, operations on mesh edges such as those incurred by the computation of convective fluxes and, on the other hand, operations on mesh tetrahedra such as those involved in the evaluation of diffusive fluxes. Both types of computations can be described as three-step sequences of the form *Gather/Compute/Scatter*.

Such kernels have been studied extensively by Gropp *et al.*[66] in the context of the parallelization of FUN3D[4] which is a CFD solver that is typical of the state of the practice at NASA. The computational kernels characterizing the numerical methods adopted in FUN3D are very similar to those discussed here. The parallelization of FUN3D has been performed in the framework of the PETSc[2] toolkit using the SPMD message-passing programming model, supplemented by multithreading at the physically shared memory level (the parallel performances of FUN3D are evaluated on both SMP systems and clusters of dual processing nodes). Particular care has been taken to obtain a scalable implementation that optimally exploits per-processor efficiency and parallel efficiency. In particular,

[1] Parallel CFD conference proceedings, see http://www.parcfd.org/general.html
[2] Portable Extensible Toolkit for Scientific Computing, http://www-unix.mcs.anl.gov/petsc/petsc-2/

single-processor performance is tuned through interlacing, blocking and edge reordering (FUN3D only involves edge-wise operations) strategies. This work received recognition in the form of a Gordon Bell Prize in the "Special Category" at the Supercomputing'99 conference[5].

2.2.1.1 Edge based computations

The evaluation of an elementary convective flux (1.12) using the numerical flux function Φ_{ij} of eq. (1.18) (or alternatively (1.19)) and taking into account the interpolated states (1.20) can be summarized as follows:

$$\left\{ \begin{array}{ll} H_{ij} & = \Phi(W_{ij}, W_{ji}, \vec{\eta}_{ij}) \approx \displaystyle\int_{\partial C_{ij}} \vec{\mathbb{F}}^c(W).\vec{\eta} dl \\ H_{ji} & = -H_{ij} \end{array} \right. \tag{2.1}$$

The elementary convective flux H_{ij} is computed at the interface between the control volumes C_i and C_j. From the point of view of the discrete equation at vertex s_i, this elementary flux contributes to a flux balance at the boundary of the control volume C_i. This flux balance involves the elementary convective fluxes computed for the control volumes C_j such that $s_j \in K(i)$ i.e. the set of neighboring vertices of s_i. Moreover, from the second of eq. (2.1), it follows that only H_{ij} needs to be computed in order to update the nodal flux balances at the two end-point of edge $e_{ij} = \{s_i, s_j\}$. Therefore, the most efficient way for evaluating the convective fluxes is to loop over the list of mesh edges and proceed as follows:

FOR each edge $e_{ij} = \{s_i, s_j\}$ of $\mathcal{T}_h$ DO

Gather $W_i = W(s_i)$ and $W_j = W(s_j)$

Gather $\vec{\nabla}\bar{W}_i = \vec{\nabla}\bar{W}(s_i)$ and $\vec{\nabla}\bar{W}_j = \vec{\nabla}\bar{W}(s_j)$

Compute the interpolated states W_{ij} and W_{ji}

Compute $H_{ij} = \Phi(W_{ij}, W_{ji}, \vec{\eta}_{ij})$

Scatter $\Phi_i = \Phi_i + H_{ij}$ and $\Phi_j = \Phi_j - H_{ij}$

ENDDO

2.2.1.2 Tetrahedron based computations

Eq. (1.22) shows that the values of the flux components $F_x^v(\tau)$, $F_y^v(\tau)$ and $F_z^v(\tau)$ used for the evaluation of the elementary diffusive flux $\Upsilon_{\tau,i}(\tau)$ are constant on the tetrahedron τ. Moreover, these values contribute to the nodal diffusive fluxes at all four vertices of tetrahedron τ. Clearly, the most efficient way for evaluating the diffusive fluxes is to loop over the list of mesh tetrahedra and proceed as follows:

FOR each tetrahedron $\tau_{ijkl} = \{s_i, s_j, s_k, s_l\}$ of $\mathcal{T}_h$ DO

Gather $W_i = W(s_i)$, $W_j = W(s_j)$, $W_k = W(s_k)$ and $W_l = W(s_l)$

Compute $F_x^v(\tau)$, $F_y^v(\tau)$ and $F_z^v(\tau)$

Compute $\Upsilon_{\tau,i}(\tau)$, $\Upsilon_{\tau,j}(\tau)$, $\Upsilon_{\tau,k}(\tau)$ and $\Upsilon_{\tau,l}(\tau)$

Scatter $\Upsilon_i = \Upsilon_i + \Upsilon_{\tau,i}(\tau)$, $\Upsilon_j = \Upsilon_j + \Upsilon_{\tau,j}(\tau)$, $\Upsilon_k = \Upsilon_k + \Upsilon_{\tau,k}(\tau)$ and $\Upsilon_l = \Upsilon_l + \Upsilon_{\tau,l}(\tau)$

ENDDO

2.2.2 Parallelisation strategy

The parallelization of the resulting flow solver relies on a classical and widely used strategy. A SPMD (Single Program Multiple Data) parallel programming model is adopted. Its implementation relies on the combination of a domain partitioning approach and a message-passing programming model. The underlying mesh is assumed to be partitioned into several submeshes, each defining a subdomain. Basically the same code is executed within each subdomain. This code is essentially the one obtained in the purely serial case augmented by several assembly phases of certain subdomain results, depending on the order of the spatial approximation and on the nature of the time advancing procedure (explicit/implicit). The assembly of the subdomain results can be implemented in separated modules and optimized for a given machine. In this study, parallel programming relies on the MPI message passing environment. This approach enforces data locality, and therefore is suitable for all parallel hardware architectures.

For the partitioning of the unstructured mesh, two basic strategies can be considered. One can verify that for the computations described herein, mesh partitions with overlapping simplify the programming of the subdomain interfacing module. However, mesh partitions with overlapping also have a drawback: they incur redundant arithmetic operations. On the contrary, non-overlapping mesh partitions incur little redundant arithmetic operations but induce additional communication steps. While assembled nodal quantities are exchanged between neighboring subdomains in overlapping mesh partitions, partially gathered quantities are exchanged and combined in non-overlapping ones. In addition, in the latter case, special care must be taken while combining the partially gathered quantities since a given vertex can belong to more than two subdomains. In summary, the parallel programming effort is maximized when considering non-overlapping mesh partitions. We refer to Farhat and Lanteri[48] for a comparison of these two approaches in the context of two-dimensional simulations. In the present study we consider both one tetrahedra wide overlapping and non-overlapping mesh partitions for second order accurate implicit computations.

2.2.3 Parallel algorithms

For an explicit time integration procedure and a one tetrahedra wide overlapping mesh partition, the main loop of the parallel flow solver can be summarized as follows:

REPEAT $step = step + 1$
- Compute the local time steps
- Compute the nodal gradients and the diffusive fluxes
- Exchange the nodal gradients
- Compute the convective fluxes
- Update the physical states
- Exchange the conservatives variables

UNTIL $step = step_{max}$

In the above pseudo-code, $step_{max}$ denotes the maximum number of time steps. For an implicit time integration procedure, the update phase is replaced by the following two-step solution procedure:

Form the implicit matrix
FOR $srl = 1$ to $nsrl$ DO
- Exchange the right-hand sides
- Perform a Jacobi relaxation

ENDDO

where $nsrl$ denotes the number of Jacobi relaxations that need to be done in order to approximately solve the linear system arising at each time step. One can notice that a global solution strategy has been selected through the choice of the Jacobi method whose parallelization is straightforward.

When a non-overlapping mesh partition is used, the explicit parallel algorithm becomes:

REPEAT $step = step + 1$

Compute the local time steps

Compute the nodal gradients and the diffusive fluxes

Exchange the partially gathered nodal gradients

Compute the convective fluxes

Exchange the partially gathered global fluxes

Update the physical states

UNTIL $step = step_{max}$

while for an implicit time integration procedure one has:

Form the implicit matrix

Exchange the partially gathered diagonal blocks of the implicit matrix

FOR $srl = 1$ to $nsrl$ DO

Exchange the partially gathered right-hand sides

Perform a Jacobi relaxation

ENDDO

We can therefore expect a lower communication cost for an implicit computation using a one tetrahedra wide overlapping mesh partition as suggested by the additional communication step involving the diagonal blocks of the implicit matrix in the non-overlapping case. Finally we note that the above pseudo-codes only show local communication steps at artificial submesh boundaries; in addition, global communication steps (reduction operations) are also necessary for the computation of the non-linear (time stepping loop) and linear residuals (linear system resolutions).

2.2.4 Automatic mesh partitioning

For the time integration procedures considered in this study, an automatic mesh partitioner should focus primarily on creating load balanced submeshes which induce a minimum amount of interprocessor communications. This can be achieved by using a two-step procedure. First, a fast and cheap partitioning scheme is used to derive an initial candidate; then, an optimization process is performed in order to realize the stated goals. While the former step consists in a global operation (the overall mesh is concerned by this step), the latter mainly concentrate on those mesh components that are neighbors of the artificial submesh interfaces (local operation). Optimization techniques that are used in this context include (among others) simulated annealing and the Kernighan-Lin algorithm. Mesh partitioning algorithms can exploit the mesh connectivity as it is the case for the Greedy algorithm (see Farhat and Lesoinne[49]). They can also be based on geometric informations such as the principal inertia directions of the mesh. In that case, a direction is first specified (one of the principal inertia directions). Next, the mesh vertices are projected (orthogonal projection) onto that direction. Finally, the projected nodes are sorted along that direction then collected to build the requested submeshes. Variations of the above algorithm can be obtained by applying the project, sort and collect paradigm in a recursive manner. More sophisticated algorithms are graph theory based such as the recursive graph bisection or the recursive spectral bisection algorithms (see Simon[128] for more details). We refer to Farhat and Lanteri[48] for an evaluation of the influence of the mesh partitioning algorithm on two-dimensional parallel simulations.

In the present study, the computational mesh is partitioned in a preprocessing step. We have used two special purpose packages that implement several mesh partitioning algorithms : `MS3D` (a Mesh Splitter for 3D applications, a description of a two-dimensional version of this preprocessor with a set of experimental results may be found in Loriot[87]) for the construction of one tetrahedra wide overlapping mesh partitions, and `TOP/DOMDEC` (a software tool for mesh partitioning and parallel processing of CSM and CFD computations [50]) to generate non-overlapping ones.

2.3 Numerical experiments

2.3.1 The selected flow calculation

The test case under consideration is the external inviscid transonic flow around a `Falcon` aircraft. The free-stream Mach number has been set to 0.85 and the angle of attack to 1°. Two unstructured tetrahedral meshes have been constructed using the `GHS3D` tetrahedral mesh generator[63]. Their characteristics are given in table 2.1. The system of Euler equations is discretized using the mixed element/volume formulation presented in section 1.2.1 of chapter 1. The linearized backward Euler implicit scheme described in section 1.4.1 of chapter 1 is used for time integration of the resulting semi-discrete equations. The convergence to steady-state is accelerated by applying a local time stepping strategy coupled to a CFL law given by CFL=$\min(i_t, 50)$ where i_t denotes the pseudo-time iteration number. At each pseudo-time step, the sparse linear system of eq. 1.84 is approximately solved using the Jacobi relaxation method with a linear threshold that has been set to $\varepsilon = 10^{-2}$. Convergence to steady state is shown on figure 2.4 in terms of the evolution of the energy residual (normalized to its initial value) versus the number of pseudo-time iterations. Selected views of the surfacic mesh of mesh FALC1 are shown on figure 2.1. Steady pressure contour lines are represented on figures 2.2 (respectively, figures 2.3) for mesh FALC1 (respectively, mesh FALC2).

Table 2.1: Characteristics of the tetrahedral mesh around the `Falcon` aircraft
N_V : # vertices - N_T : # tetrahedra - N_F : # boundary faces

Mesh	N_V	N_T	N_F
FALC1	130,478	768,438	12,552
FALC2	1,035,670	6,147,504	50,208

2.3.2 Computing platforms and notational conventions

Numerical experiments have been performed on several clusters of PCs:

- a cluster consisting of 19 dual nodes Pentium III/933 Mhz, with 512 Mb of SDRAM 133 Mhz each, interconnected by an Ethernet 100 Mbit/s switch.
- a cluster of 16 dual nodes Pentium IV/2 Ghz with 1 Gb of RDRAM 800 Mhz each, interconnected by an Ethernet 1 Gbit/s switch.

All these clusters are running the Linux operating system. Performance results are given for 64 bit arithmetic computations. The code is written in Fortran 77 and parallel programming relies on the MPICH implementation of MPI. In the following tables and discussions, N_p is the number of processes for the parallel execution (N_p also represents the number of subdomains), N_n is the number of processing nodes (since most of the clusters described previously are based on dual boards, this figure will allow to make the distinction between calculations using one or two processes on a given node in order to assess the impact of the local memory architecture). Moreover, "Total time" denotes the total (wall clock) simulation time and "CPU time" stands for the corresponding total CPU time taken as the maximum of the per process values. Finally, "%CPU" is the ratio of the total CPU time to the total wall clock time. The parallel speedup $S(N_p)$ is always calculated using the total wall clock times.

2.3.3 Parallel performance results

Parallel performance assessment of the underlying unstructured mesh flow solver is studied with respect to the partitioning strategy. We first discuss results of numerical experiments that have been performed using mesh FALC1. Tables 2.2 and 2.3 summarize timings obtained on the cluster of Pentium III/933 Mhz processors and Ethernet 100 Mbit/s interconnection. Corresponding timings on the cluster of Pentium IV/2 Ghz processors and

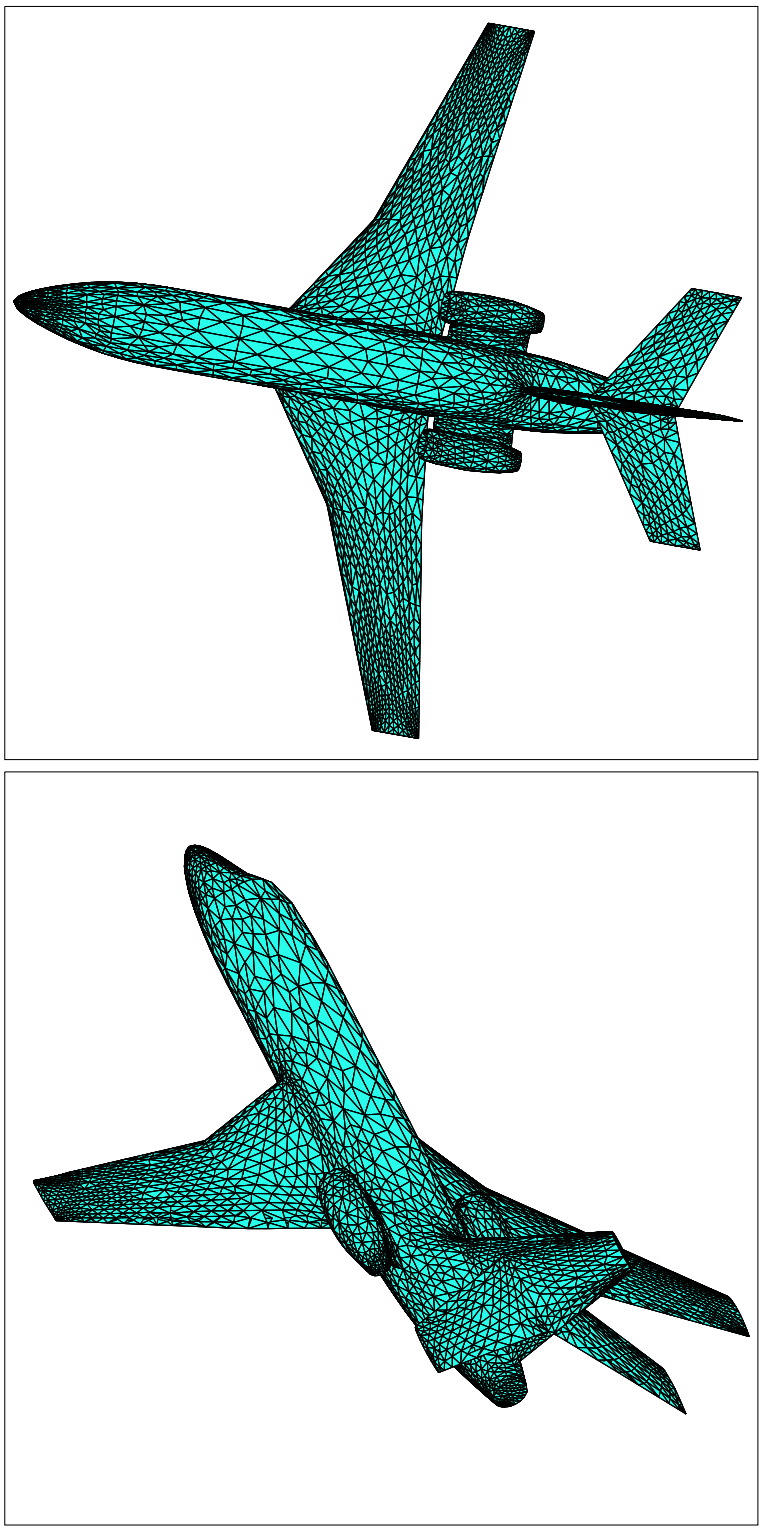


Figure 2.1: External flow around a `Falcon` aircraft: surfacic mesh (mesh FALC1)

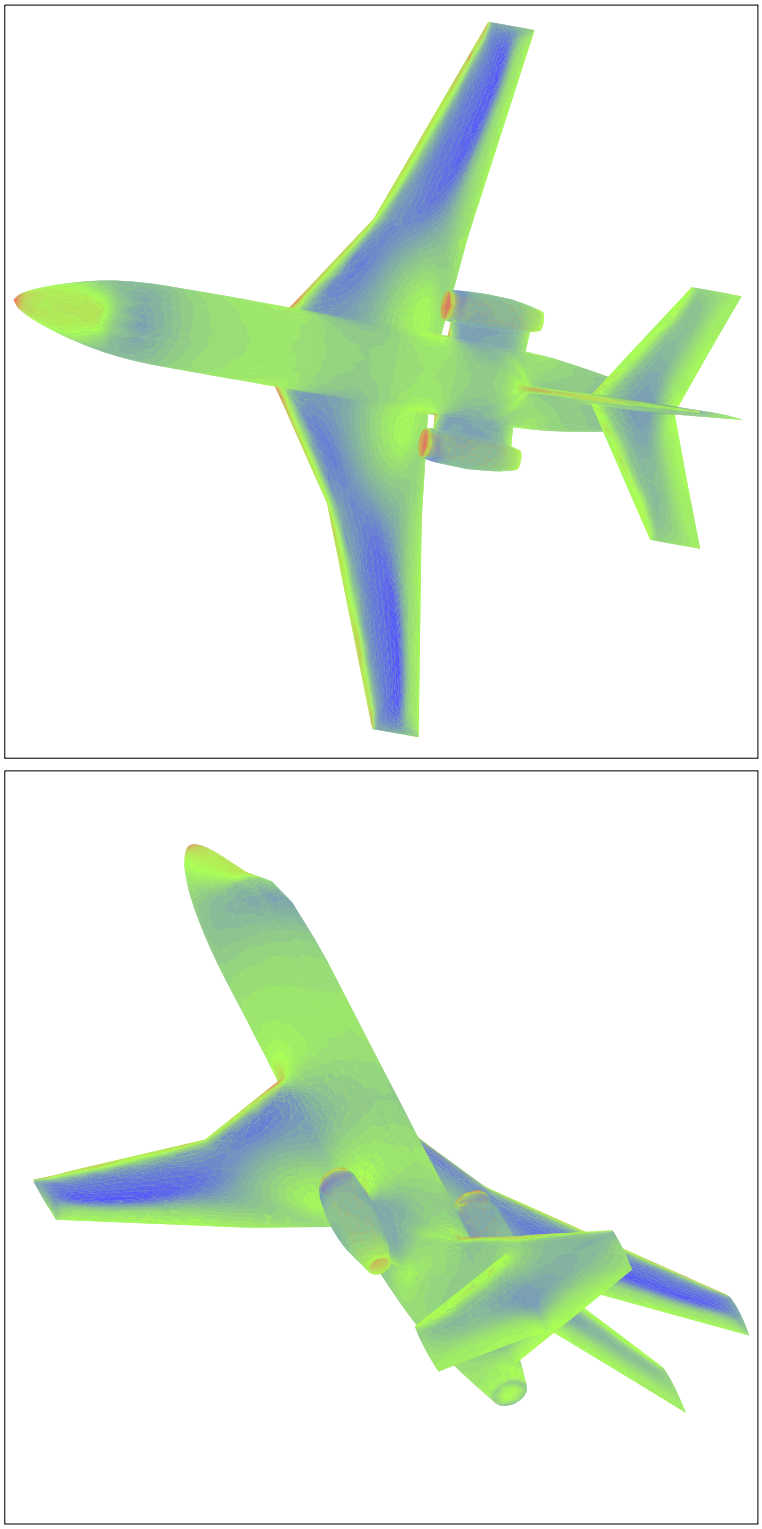


Figure 2.2: External flow around a `Falcon` aircraft
Steady pressure contour lines on the aircraft (mesh FALC1)

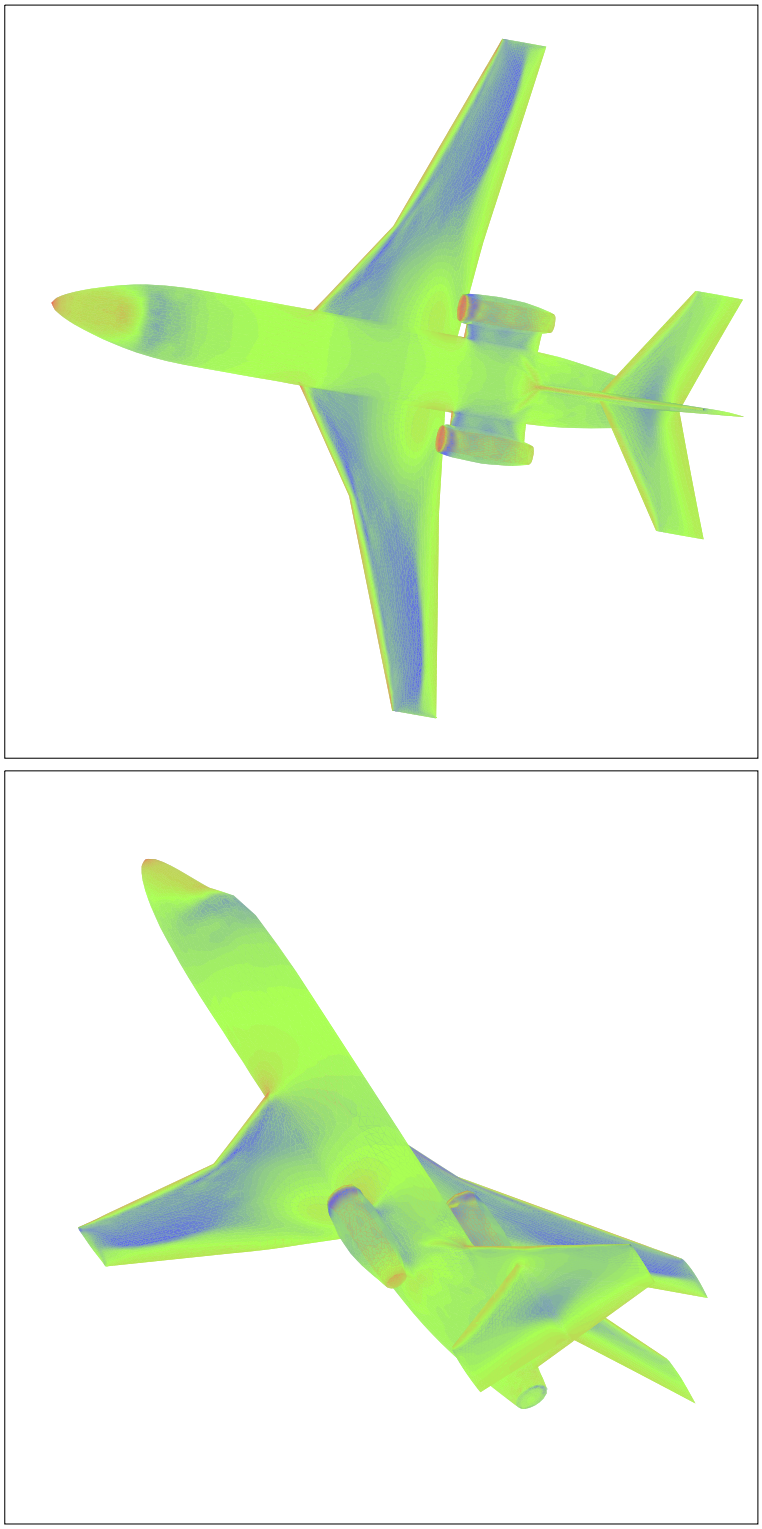


Figure 2.3: External flow around a `Falcon` aircraft
Steady pressure contour lines on the aircraft (mesh FALC2)

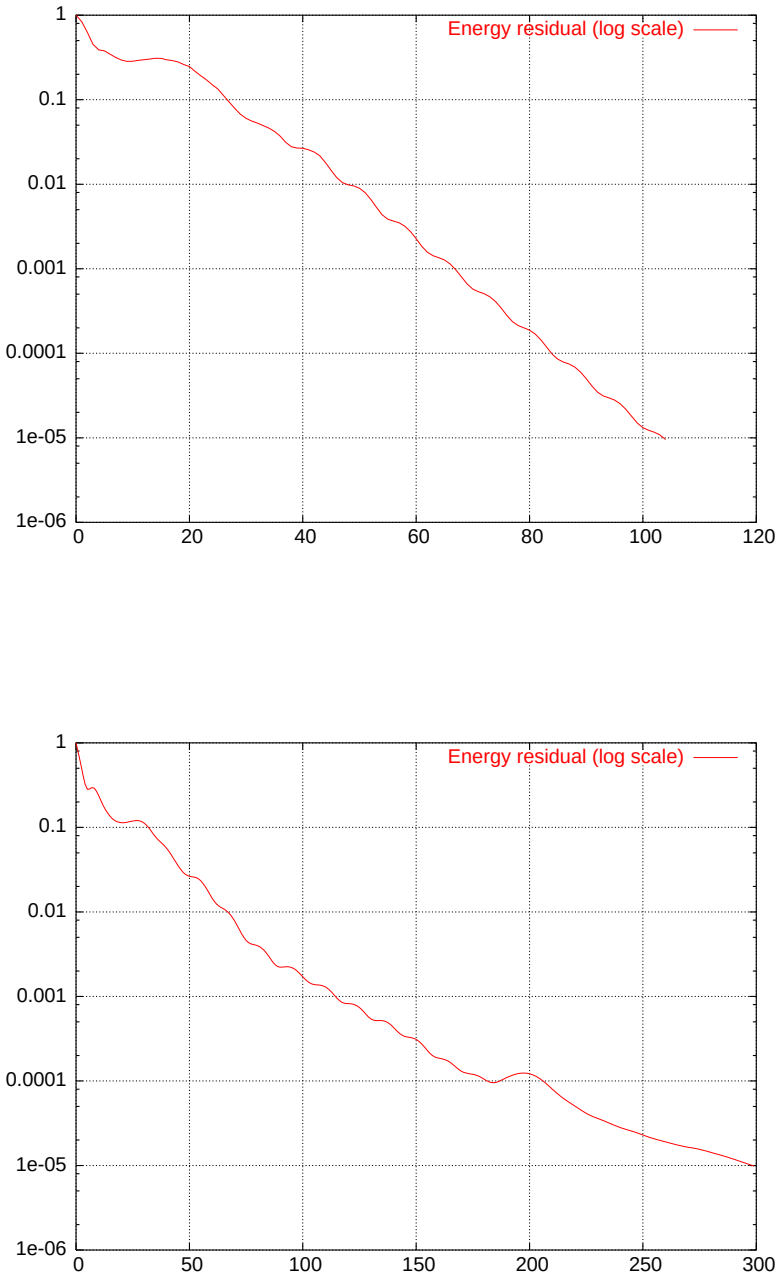


Figure 2.4: External flow around a `Falcon` aircraft
Convergence to steady-state (top: mesh FALC1 - bottom : mesh FALC2)

Ethernet 1 Gbit/s interconnection are given in tables 2.4 and 2.5. Note that comparisons are made on the basis of the number of processes per processing node. These results call for several remarks:

- parallel efficiency as measured by the "%CPU" ratio is not sensitive to the partitioning strategy as far as the number of processes is lesser than 32. For $N_p = 32$, the largest difference is observed on the cluster of Pentium IV/2 Ghz processors and Ethernet 1 Gbit/s interconnection (i.e. 76% for overlapping partitions and 83% for non-overlapping partitions). This is somewhat surprising since, as mentioned in subsection 2.2.2, the main disadvantage of overlapping partitions over non-overlapping ones is redundant arithmetic operations and one can reasonably expect that the effect of the latter should be less remarkable on the Pentium IV processor. Therefore, the reason for this better parallel efficiency on the cluster of Pentium IV/2 Ghz processors must be found elsewhere. Indeed, a possible explanation is given by figures 2.5 and 2.6 that represent the effective numbers of interface vertices for the 32 subdomain overlapping and non-overlapping partitions. For the overlapping partition, one must make a distinction between vertices that are sent and those that are received. These figures illustrate another advantage of the non-overlapping partitioning strategy: as the number of subdomains increases, the size of artificial interfaces can be reduced more efficiently. In the present case, despite the fact that the parallel flow solver based on the non-overlapping partitioning strategy is characterized by a higher communication load, the smaller interface sizes (by a factor 1.9 in the average) translates in a lower communication cost and thus, a higher parallel efficiency.

- The parallel speed-up is consistently better for the parallel flow solver based on the non-overlapping partitioning strategy. This is essentially the result of the fact that redundant arithmetic operations are minimized in that case. On the contrary, for overlapping partitions, the aggregate number of arithmetic operations is larger than the one that can be deduced the global (not-partitioned) mesh. Thus, in some sense, the parallel speed-up as evaluated here compares executions times associated to problems of different size.

- Clearly, the multi-threading execution mode is not a good option on the Pentium III/933 Mhz based system for the kind of computational kernels considered here. For $N_p = 4$, the single-process per node execution mode yields an execution time which is about half the execution time obtained when two processes are run on the same node, independently of the partitioning strategy. For instance, the ratio between the execution times obtained for $N_n = 2$ and $N_n = 4$ is equal to $4995/2766 = 1.8$ for overlapping partitions and $5075/2795 = 1.8$ for non-overlapping partitions. On the cluster of Pentium IV/2 Ghz processors, the corresponding figure is equal to $1501/1399 = 1.1$ ($1527/1062 = 1.4$ for non-overlapping partitions). This difference in behavior between the two clusters is for a great part related to the shared memory sub-system characteristics (type of RAM, speed of the memory bus). In particular, these characteristics can have a dramatic impact on the treatment of the indirect addressing (gather/scatter) operations as experienced here on the Pentium III/933 Mhz based system. For information, the floating-point rate which is achieved on the cluster of Pentium IV/2 Ghz processors for $N_p = 16$ and non-overlapping partitions is equal to 2.3 Gflop/s for $N_n = 8$ and 3.2 Gflop/s for $N_n = 16$.

Table 2.2: External flow around a `Falcon` aircraft (mesh FALC1)
Cluster of Pentium III/933 Mhz (Ethernet 100 Mbit/s interconnection)
Overlapping partitions

N_p	N_n	CPU time	Total time	% CPU	$S(N_p)$
4	2	4858 sec	4995 sec	97.5%	1.00
16	8	1272 sec	1485 sec	85.5%	3.90
32	16	757 sec	1018 sec	74.5%	5.00
4	4	2635 sec	2766 sec	95.0%	1.00
16	16	722 sec	834 sec	86.5%	3.65

Table 2.3: External flow around a `Falcon` aircraft (mesh FALC1)
Cluster of Pentium III/933 Mhz (Ethernet 100 Mbit/s interconnection)
Non-overlapping partitions

N_p	N_n	CPU time	Total time	% CPU	$S(N_p)$
4	2	4935 sec	5075 sec	97.5%	1.00
16	8	1242 sec	1387 sec	89.5%	4.00
32	16	672 sec	898 sec	75.0%	5.70
4	4	2653 sec	2795 sec	95.0%	1.00
16	16	734 sec	828 sec	88.5%	3.60

Table 2.4: External flow around a `Falcon` aircraft (mesh FALC1)
Cluster of Pentium IV/2 Ghz (Ethernet 1 Gbit/s interconnection)
Overlapping partitions

N_p	N_n	CPU time	Total time	% CPU	$S(N_p)$
4	2	1463 sec	1501 sec	97.5%	1.00
16	8	366 sec	398 sec	92.0%	4.00
32	16	184 sec	243 sec	76.0%	6.20
4	4	1356 sec	1399 sec	97.0%	1.00
16	16	347 sec	372 sec	93.5%	3.90

Table 2.5: External flow around a `Falcon` aircraft (mesh FALC1)
Cluster of Pentium IV/2 Ghz (Ethernet 1 Gbit/s interconnection)
Non-overlapping partitions

N_p	N_n	CPU time	Total time	% CPU	$S(N_p)$
4	2	1494 sec	1527 sec	98.0%	1.00
16	8	345 sec	380 sec	91.0%	4.35
32	16	167 sec	201 sec	83.0%	7.60
4	4	1037 sec	1062 sec	97.5%	1.00
16	16	243 sec	266 sec	91.5%	4.00

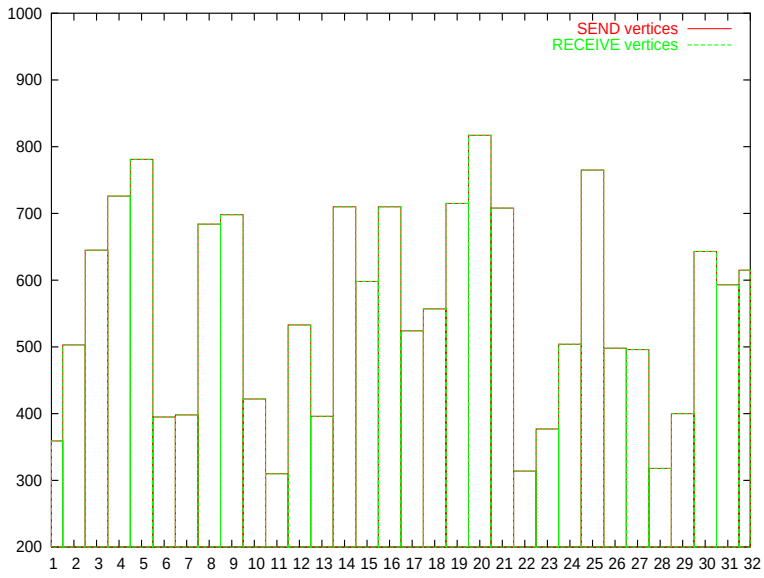


Figure 2.5: External flow around a `Falcon` aircraft (mesh FALC1)
Number of send/receive vertices for the 32 subdomain overlapping partition

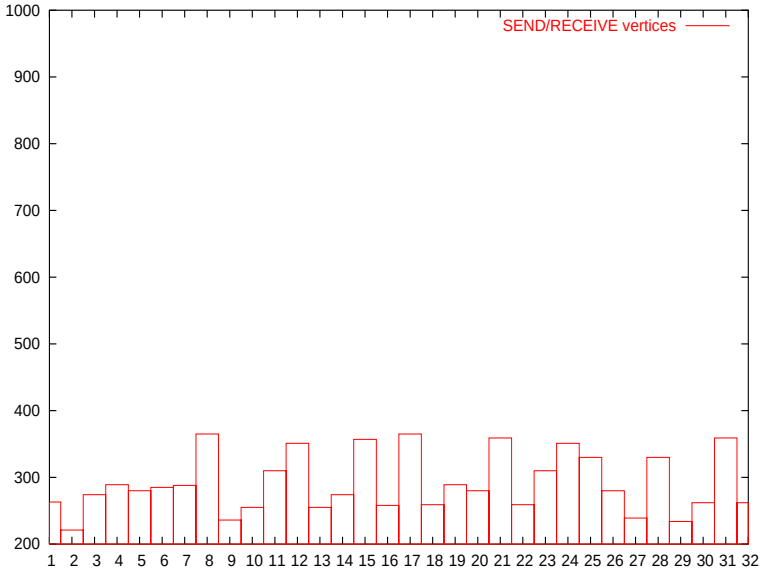


Figure 2.6: External flow around a `Falcon` aircraft (mesh FALC1)
Number of send/receive vertices for the 32 subdomain non-overlapping partition

We conclude this section with performance results obtained for calculations based on mesh FALC2. These results are summarized in table 2.6. Figure 2.7 represents the effective numbers of vertices for the 32 subdomain overlapping and non-overlapping partitions. For the overlapping partition, the average number of vertices is 39382 while it is equal to 34756 for the non-overlapping partition that is, approximately 11% less. However, the corresponding reduction of computing time is only of 4%. Clearly, we are lacking here some performances results for a higher number of subdomains (e.g. $N_p = 64$) in order to evaluate parallel speedup figures and demonstrate with more evidence the superiority of the approach based on non-overlapping partitions.

Table 2.6: External flow around a `Falcon` aircraft (mesh FALC2)
Cluster of Pentium IV/2 Ghz (Ethernet 1 Gbit/s interconnection)
Overlapping versus non-overlapping partitions ($N_p = 32$ and $N_n = 16$)

Overlap	CPU time	Total time	% CPU
With	5024 sec	5415 sec	92.8
Without	4810 sec	5184 sec	92.8

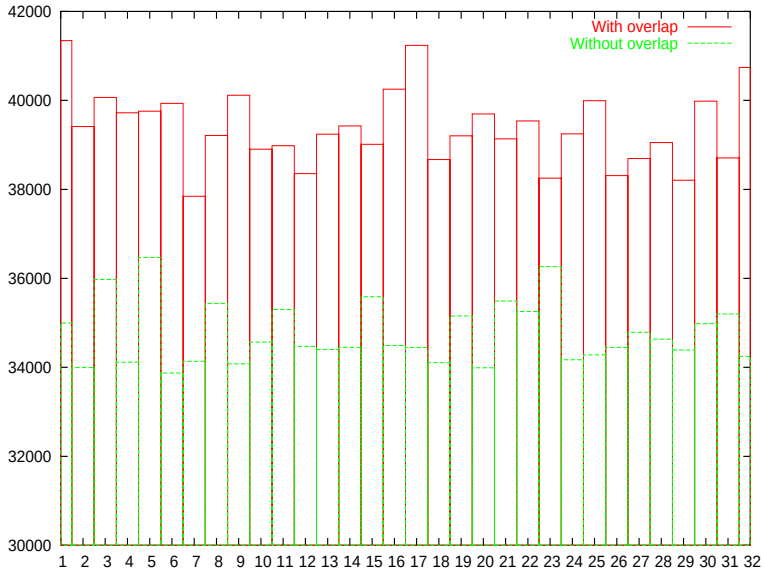


Figure 2.7: External flow around a `Falcon` aircraft (mesh FALC2)
Number of vertices for the 32 subdomain overlapping and non-overlapping partitions

2.4 Conclusion

In this chapter we have studied the parallelization of a CFD solver for the Euler and Navier-Stokes equations that implements the mixed element/volume formulation on unstructured tetrahedral meshes described in section

1.2.1 of chapter 1. The parallelization of this solver relies on a classical and widely used SPMD approach that combines a partitioning of the underlying mesh and a message-passing programming model. Based on lessons drawn from previous works (Fezoui and Lanteri[53], Farhat and Lanteri[48], Fezoui *et al.*[55]), we have considered two implementations of this strategy. The first one makes use of overlapping mesh partitions; it contributes to minimize the programming effort on the original serial algorithm but is also characterized by redundant arithmetic operations. The second strategy uses non-overlapping mesh partitions and demands additional programming effort. The main conclusion of the experimental results reported here is that the approach based on non-overlapping mesh partitions demonstrates better parallel performances (especially in terms of the parallel speed-up). The superiority of this approach is made clearer as the number of subdomains is increased from 16. Despite the fact that this approach is characterized by higher communication loads (additional communication steps, increased message sizes due to the exchange of partially gathered quantities instead of nodal values), the reduction of pure computational times through the elimination of redundant arithmetic operations plays the major role in this result.

Chapter 3

Parallel multigrid methods

In this chapter, we describe our contributions concerning the design of parallel, linear and non-linear, multigrid methods on unstructured finite element meshes for the calculation of 2D and 3D compressible flows.

3.1 Parallel multigrid by volume agglomeration

3.1.1 Introduction

One of the most remarkable innovations of the last 25 years in the computational fluid dynamics field has been the progressive introduction of unstructured mesh discretization techniques in the simulation tools used in various transport and energy related industrial sectors (e.g. aeronautical, automotive, space and electrical power supply). The fluid flows that characterize the applications of interest to these domains call for complex physical models (turbulence, combustion, moving computational domains, etc.) around or inside irregularly shaped geometries. Finite element or finite volume approximation methods relying on fully unstructured meshes are particularly well suited to these situations. Unstructured meshes present two main advantages: they allow for an accurate discretization of all the details of the geometry shape (e.g. aircraft surface, combustion engine diffuser, etc.) and, they can be locally refined and dynamically adapted or deformed to take into account the complex and unsteady features of the underlying flow. Several major contributions in this domain have been concerned with the design of finite volume type methods involving sophisticated upwind solvers to account for the dominant hyperbolic nature of the underlying flow models[120]-[131]-[110]. Such numerical schemes are very robust with respect to the treatment of some characteristics of the system of Euler equations, such as shock waves and contact discontinuities, which constitutes the basic model of compressible fluid mechanics. Nevertheless, this robustness is obtained at the expense of a numerical viscosity which is naturally attached to upwind schemes. In most of the cases, the loss of accuracy resulting from the introduction of this numerical diffusion does not allow for a correct characterization of the main flow features. Then, the strategy generally adopted to increase the accuracy of the approximation consists in using upwind schemes based on the MUSCL (Monotonic Upwind Schemes for Conservation Laws) technique initially proposed by Van Leer[141] and extended to unstructured meshes by Fezoui[51], in conjunction with TVD (Total Variation Diminishing) formulations[69]-[135]-[132]-[56]-[109] in order to compute solutions that are physically correct. For the time integration of the resulting semi-discrete equations, several works have been concerned with the design of linearized implicit schemes for the calculation of steady flows[56]-[54] and unsteady flows[89].

For what concerns the resolution of the associated algebraic systems, the main question still appears to be the efficiency of unstructured mesh solvers compared with structured ones (e.g. multi-block methods). Many efficient methods developed in the structured context are not easily extensible to unstructured meshes and much research work has still to be done in this direction. It seems clear that these studies must target the design of iterative methods that are (1) of optimal complexity (i.e. whose convergence is independent of the number of degrees of freedom) and (2) well adapted to modern parallel computing platforms. Concerning the first point, a promising

research direction aims at proposing resolution methods that are based on a certain hierarchical treatment of the underlying data. Multigrid methods belong to this framework[68]-[18]-[143]. Multigrid methods can be classified according to several criteria. A first distinction comes from the nature of the system to be solved i.e. linear versus non-linear system. The FAS (Full Approximation Scheme) method is the most general formulation of a multigrid algorithm. It has been first proposed by Brandt[14]-[15] for the resolution of non-linear systems. In the case of linear systems, the FAS algorithm can be easily reformulated to yield a linear multigrid algorithm. However, in both cases, the basic principle consists in computing corrections on a set of coarse discretization levels which are then used to update the solution computed on the finest discretization level. Moreover, this basic principle can be coupled to a grid refinement technique yielding the so-called FMG (Full MultiGrid) algorithm.

The basic components of any multigrid method are: (1) a hierarchy of discretization levels and, for each level, the corresponding formulation of the system to be solved, (2) a set of operators for transferring data between two consecutive levels of the hierarchy, (3) a smoothing method used to damp the high frequency components of the iterative error and, (4) a method to solve the system formulated on the coarsest discretization level. The definition of the hierarchy of discretization levels allows to further separate multigrid methods in two families:

- for $geometric$ multigrid methods, the hierarchy of discretization levels consists of a set of meshes (or grids) of increasing (or decreasing) resolution. The finest grid is the support of the targeted solution. The coarser grids are used to accelerate the convergence of the iterative process. On each coarse grid, a discrete representation of the original system of PDEs is obtained either by using the approximation method adopted on the finest grid or, by applying a purely algebraic technique involving the discrete operator on a finer grid and inter-grid transfer operators[18].

- $algebraic$ multigrid methods[138] take as input the algebraic system resulting from the discretization of the original system of PDEs on the finest grid. They do not require coarser grids as it is the case for $geometric$ multigrid methods. Instead, coarser formulations of the fine grid algebraic system and associated inter-grid transfer operators are obtained through the use of aggregation techniques that are directly applied to matrix coefficients. The application of $algebraic$ multigrid methods is essentially restricted to linear systems.

In the case of $geometric$ multigrid methods, the construction of the hierarchy of discretization levels in conjunction with unstructured meshes is not an easy task and has been the subject of several works. A first approach consists in applying a recursive refinement algorithm (i.e. by subdividing mesh elements) to a coarse discretization of the computational domain. It is clear that such a strategy yields a fully nested hierarchy of discretization levels. Alternatively, a multigrid method based on not-nested discretization levels has been developed by Mavriplis and Jameson[97]-[98]. More recently, Guillard[67] has proposed a coarsening method that builds a coarse mesh from a finer one by deleting selected mesh vertices and applying a re-meshing algorithm.

These various approaches have allowed the construction of robust and efficient algorithms for several domains of application. However, the fact that they require handling several meshes for the same application make them difficult to exploit in an industrial context. One way to overcome this restriction consists in using a sequence of coarse levels that are automatically built from the finer discretization grid. One strategy of this type makes use of topological relations of a dual mesh such as the control volume mesh associated to the mixed finite element/volume formulation considered in section 1.2.1 of chapter 1. This particular approach results in the so-called multigrid by volume agglomeration method that has been first developed by Lallemand $et\ al.$[82] at INRIA Sophia Antipolis, for the resolution of the system of two-dimensional Euler equations. A few years later, in the United States, more precisely at ICASE at the NASA Langley research center, this method has been the subject of several important contributions due to Mavriplis and co-workers[95]-[96]-[92]-[93]-[94]. Meanwhile, the Sinus team at INRIA Sophia Antipolis has also produced several contributions concerning the extension and the application of the multigrid by volume agglomeration method to laminar and turbulent viscous flows in the 2D case[81]-[23]-[57]. However, these contributions are essentially the results of basic research activities and the methodology has rarely been applied to more realistic (i.e. industrial) situations. In particular, the application of the multigrid by volume agglomeration method to the calculation of complex 3D flows as well as its adaptation to modern parallel computing platforms have never been considered simultaneously.

The main contribution of the present work is twofold: on one hand, we demonstrate the successful extension and application of the multigrid by volume agglomeration principle to the acceleration of complex three-dimensional flow calculations on unstructured tetrahedral meshes and, on the other hand, we enhance further the overall efficiency of the methodology through its adaptation to distributed memory parallel computing platforms. Moreover, a non-trivial aspect of this work is that the corresponding software developments are taking place in an existing industrial flow solver. This activity was supported by a consortium consisting of three industrial end-users: EDF, SNECMA and RENAULT. Then, a major objective of the present study was to demonstrate the potentiality of agglomerated multigrid for flow calculations relevant to these industrial partners. For this purpose, the proposed multigrid methodology is applied to two representative turbulent steady flow calculations within complex geometries.

The underlying industrial flow solver is based on an averaged form of the multi-component Navier-Stokes equations coupled to a $k - \varepsilon$ turbulence model. The spatial discretization combines finite element and finite volume concepts and is designed on unstructured tetrahedral meshes. Steady state solutions of the resulting semi-discrete equations are obtained using an Euler implicit time advancing strategy which is based on the following features : *linearization* (approximate linearization of the convective fluxes and exact differentiation of the viscous terms), and *local time stepping and CFL law* (a local time step is computed on each control volume). Then, each pseudo-time step can require the solution of as many as three sparse linear systems for the mean flow variables, the turbulent variables and the chemical species. In the existing solver, these systems are (approximately) solved using several sweeps of a relaxation method (that is Jacobi or Gauss-Seidel methods). Here, parallel linear multigrid algorithms by volume agglomeration are developed for the acceleration of the solution of these linear systems. Then, the basic relaxation methods are used as smoothers for the proposed algorithms. The standard agglomerated multigrid makes use of an isotropic *greedy* type coarsening algorithm for the automatic construction of the coarse grid discretizations.

The rest of this section is organized as follows. In subsection 3.1.2 we describe the main features of the underlying industrial flow solver which are of interest to our study. In subsection 3.1.3 we outline the main ingredients of the linear multigrid by volume agglomeration method which is at the heart of our study. In this section we only consider the standard (isotropic) coarsening strategy for the construction of the coarse discretizations of the computational domain. Subsection 3.1.4 discusses the parallelization aspects. In subsection 3.1.5, results are presented for steady flow calculations performed on a SGI Origin 2000 MIMD system. Section 3.1.6 is concerned with the improvement of the efficiency of the standard agglomerated multigrid algorithm through directional coarsening. This last aspect is treated here as a feasibility study as the resulting multigrid algorithm is only applied to one of the cases considered in section 3.1.5 and not to the most appropriate calculations (that is, to calculations involving adapted or deforming meshes). Finally, subsection 3.1.7 concludes this study.

3.1.2 The N3S-NATUR flow solver

The starting point flow solver is the N3S-NATUR software package which is the result of a coordinated effort for the development of a parallel solver dedicated to the numerical simulation of industrial compressible steady or transient flows. This activity was supported by a consortium consisting of three industrial partners which are end-users and co-developers of N3S-NATUR (EDF, Snecma and Renault), two software companies (Simulog and Metraflu) and two research institutions (Ecole Centrale de Lyon and INRIA). The main characteristics of N3S-NATUR are given below:

a) physical features : N3S-NATUR is currently able to compute laminar or turbulent flows governed by the Navier-Stokes equations. Turbulence modeling is based on a two-equation $k - \epsilon$ model coupled with special wall boundary conditions to simulate boundary layers. Multi-component flows can be simulated with a modeling of the molecular diffusion. N3S-NATUR can handle 2D and 3D arbitrary complex geometries and is able to compute both confined and external flows. This software is particularly well suited to strong shocks evaluation such as those found in aeronautics, and behaves very well for a wide range of Mach numbers ($0.1 < M < 7.0$).

b) boundary conditions : several types of boundary conditions can be considered including periodicity between parallel or non parallel faces, wall boundary conditions (slip condition, wall law, thermal exchange), inflow and outflow conditions, compatibility relations (for inflow and outflow).

c) numerical features : the multi-component Navier-Stokes equations are solved in conservative form. The discretization in space relies on a mixed finite volume (for the convective terms)/finite element (for the diffusive terms) formulation which is briefly recalled hereafter. Unsteady flows based on deforming meshes can be handled thanks to an appropriate calculation of the convective terms. Explicit or linearized implicit time integration techniques are available. Jacobi and Gauss-Seidel relaxations are implemented for the solution of the linear systems resulting from the implicit time integration scheme.

N3S-NATUR is applied to the numerical simulation of flows that are modeled by the 3D compressible Navier-Stokes equations for turbulent multi-component flows. These equations are discretized on unstructured tetrahedral meshes using the mixed element/volume formulation described in section 1.2.1 of chapter 1 with some particular adaptations for what concern the equations associated to the chemical components. Finally, the semi-discrete equations are time integrated using the linearized backward Euler implicit scheme presented in section 1.4.1. Then, each pseudo-time step can require the solution of as many as three sparse linear systems for the mean flow variables, the turbulent variables and the chemical species. In N3S-NATUR, these systems are (approximately) solved using several sweeps of a relaxation method (that is Jacobi or Gauss-Seidel methods).

3.1.3 Linear multigrid by volume agglomeration

3.1.3.1 Motivations

There are two practical situations in the N3S-NATUR solver that call for the solution of large sparse linear systems:

- the linearized implicit time integration scheme. For the multi-component Navier-Stokes equations augmented by a two-equation $k-\varepsilon$ turbulence model, as many as three systems need to be solved : one for the mean flow variables, one for the turbulent variables and one for the chemical species. This results from the fact that a loosely coupled approach is used in the implicit treatment of the full system of partial differential equations;

- the mesh update procedure in the case of an unsteady calculation on a deforming geometry[109]. This aspect is not considered in this study (the application of multigrid acceleration in this case is the object of an ongoing effort).

In realistic applications such as those that can be solved with the N3S-NATUR solver, the linear system solution steps using Jacobi or Gauss-Seidel relaxations, generally represent between 80% and 90% of the total simulation time.

The main advantages of using Jacobi or Gauss-Seidel relaxations are that they do not require additional temporary storage (as it is the case for GMRES for example) which is a criterion of crucial importance for an industrial software package, they do not need to be preconditioned, and they are naturally parallelizable (at least for the Jacobi method; for Gauss-Seidel, a mixed formulation can be adopted, where Jacobi relaxations are used for unknowns that are shared by several processors). However, the main drawback is that these solvers are not numerically efficient (especially Jacobi) compared to preconditioned Krylov methods. The present work is concerned with the development and evaluation of parallel linear multigrid algorithms by volume agglomeration in order to allow for an efficient treatment of the linear system solution steps of N3S-NATUR.

3.1.3.2 Standard coarsening algorithm

The multigrid method adopted here is based on the use of macro elements (macro control volumes) that form the coarse discretizations of the computational domain. It is an extension of the linear multigrid approach developed by Mulder[105] and Lallemand $et\ al.$[82] to accelerate the solution of linear systems. In [82] the adopted coarsening

algorithm is based on neighboring relations. Starting from a fine unstructured discretization, one wants to generate a hierarchy of coarse levels. The standard approach makes use of an isotropic *greedy* type coarsening algorithm that assembles neighboring control volumes of the finest grid (e.g. those having a common boundary) to build the macro elements of the coarser level.

We denote by G_k with $k = 1, \cdots, K$ the grid hierarchy. Each grid level G_k is formed of control volumes C_i^k and we denote by N_i^k denotes the set of indices of the macro control volumes that are neighbors of C_i^k. The algorithm is given below and is illustrated on figure 3.1 in the two-dimensional case.

`Algorithm 1` *Standard grid coarsening algorithm by volume agglomeration.*

`[0]` *Set* $j = 0$ *(*j *stands for the counter of coarse grid control volumes)*

`[1] LOOP` *on each control volume* C_i^k *of grid level* G_k

`[2] IF` C_i^k has already been included in a group $C_q^{k+1} \quad q = 1, \cdots, j-1$ `THEN`

`[2a]` Consider the next control volume i.e. `GOTO [1]`

`ELSE`

`[2b]` Set $j \leftarrow j+1$ and create a new group C_j^{k+1} containing C_i^k

`[2c]` Build the coarse grid control volume C_j^{k+1}:

$$C_j^{k+1} = C_i^k \bigcup_{p \in N_i^k} C_p^k \quad \text{such that} \quad C_p^k \notin C_q^{k+1}, \quad q = 1, \cdots, j-1$$

`[2d] GOTO [1]`

`[2 end] ENDIF`

`[1 end] ENDLOOP`

The main advantage of this method is that it allows for an automatic generation of the coarser discretizations without building any coarse tetrahedrization.

3.1.3.3 Coarse grid approximations of the convective and diffusive terms

We recall that the finite volume formulation adopted on the finest mesh is such that an elementary convective flux is computed at the interface between two control volumes. In the multigrid by volume agglomeration method, they are computed in the same way on a coarse level, between two macro elements. Note that on the coarse grids, this computation is limited to first order accuracy because nodal gradients cannot be evaluated as they are on the finest mesh (in other words, the MUSCL method cannot be applied on the coarse levels). Nevertheless, this fact does not really represent a problem here since the multigrid method is used to accelerate the solution of a linear system whose Jacobian matrix is based on the linearization of a first order convective flux. Both conservative variables and normal vectors are interpolated between the different grids. The coarse grid variables are deduced by inter-grid transfer operators . The normal vectors, linked with each coarse macro control volume, result from the summation of the finer grid vectors (associated to the fine mesh control volumes that have a common boundary with the macro elements) ; as a result, one flux at most is computed between two macro control volumes. To evaluate the diffusive laminar and turbulent terms on a coarse level, related basis functions are needed. Indeed, in the finite element formulation on the fine grid, the equations are integrated and assembled by edges (convective terms) and tetrahedra (diffusive terms). As tetrahedra do not exist on the coarser grids, it is necessary to define a new formulation for the calculation of diffusive terms; we refer to Carré[23] for a more detailed description of the adopted strategy.

3.1.3.4 Inter-grid transfer operators

A condition to obtain multigrid efficiency is that the summation of the orders of the inter-transfer operators is greater than the order of the partial differential equation to be solved. In order to solve the Navier-Stokes equations, this condition, developed in [143] and [70], requires that either prolongation or restriction be linear. However, a linear interpolation is not easily built in an agglomeration context. In our case, we keep the same order for both restriction and prolongation operators which is in accordance with the previous condition only for the convective approximation, but allows simple and diagonally dominant coarse grid matrices to be built. These operators are discussed in more details in the paragraph 3.2.2.4.1 of section 3.2.

3.1.4 Parallel computing issues

3.1.4.1 Parallelization of the monogrid solver

The parallelization of the monogrid version of N3S-NATUR is studied in Lanteri and Loriot[85]. A SPMD (Single Program Multiple Data) parallel programming model is adopted. Its implementation relies on the combination of a domain partitioning approach and a message-passing programming model. This strategy is discussed in more details in section 2.2.2 of chapter 2. Here, we simply recall that for the partitioning of the unstructured mesh, two basic strategies can be considered. The first one is based on the introduction of an overlapping region at subdomain interfaces (this option corresponds to a *vertex-wise* partitioning) and is particularly well suited to the mixed finite element/volume formulation considered in this study. The second possible strategy is based on a non-overlapping mesh partition (this option corresponds to an *element-wise* partitioning). Such a strategy complicates the parallel implementation of the mixed finite element/volume formulation since partially gathered quantities need to be handled and exchanged at submesh interfaces. However, it has been our experience that parallel performances are better with this second strategy. Indeed, the strategy based on overlapping mesh partitions induced redundant floating-point operations that have a negative impact on the parallel speedup when the number of submeshes is increased (see [84] for more details). In [85] simplicity of programming has been preferred due to the complexity of the underlying software. Thus, the parallelization of the monogrid version of N3S-NATUR relies on the use of a one tetrahedron-wide overlapping zone between neighboring submeshes.

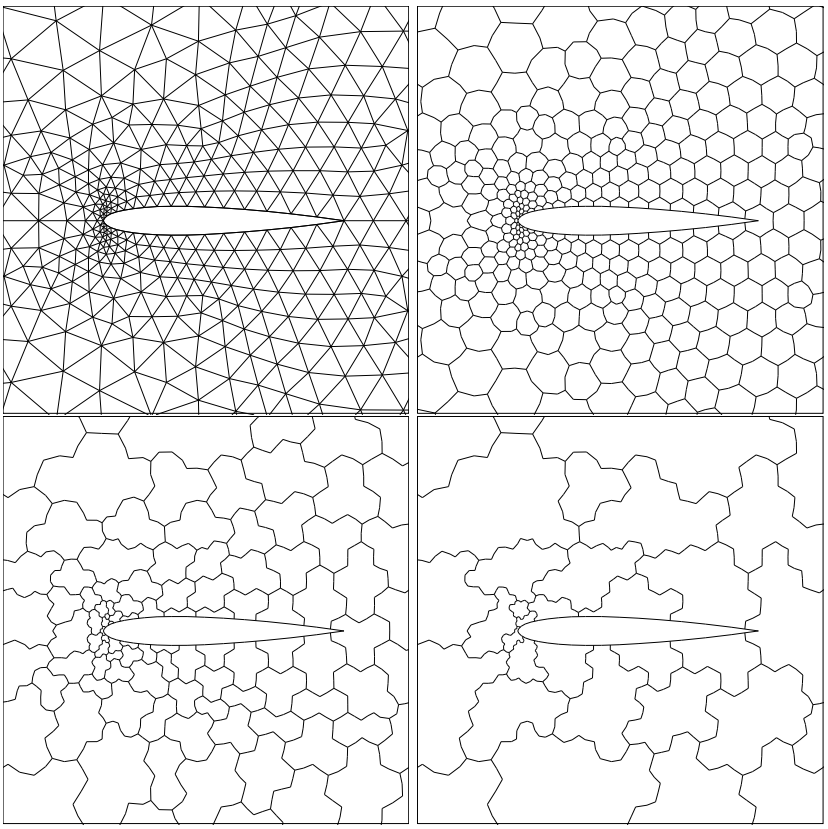


Figure 3.1: Illustration of the grid coarsening by agglomeration
Top left : triangular grid - Top right : dual grid
Bottom left : first agglomerated level
Bottom right : second agglomerated level

3.1.4.2 Parallelization of the multigrid solver

The parallelization of the multigrid method described previously essentially consists in an extension of the strategy adopted for the monogrid version of N3S-NATUR. In practice, the operations on the coarse grid levels are distributed in the same way as they are on the finest (tetrahedral) mesh. We note that this strategy corresponds to applying an *intra-level* parallelization. Such a strategy requires a preliminary step aiming at the construction of partitioned coarse grid levels. In this study, this has been achieved by developing a parallel variant of the original *greedy* type coarsening algorithm 1 which now includes additional communication steps for a coherent construction of the communication data structures on the partitioned coarse grid levels. The choice of developing a fully parallel coarsening algorithm rather than building a sequential (and probably un-coupled) preprocessing tool, is mainly motivated by the future utilization of the resulting solution strategy in the context of deforming or adaptive meshes. As stated previously, the parallelization on the monogrid solver is based on the use of an overlapping mesh partition. The overlapping zone is one-tetrahedron wide. In order to build the additional communication data structures that are required for the parallelization of coarse grid operations, we make use of the notion of submesh ownership. We consider that a submesh is the owner of all the mesh vertices it contains except those that are placed on the exterior side of artificial boundaries (referred below as interface vertices). By owner we mean that the submesh is responsible for the final update of the physical value attached to the corresponding vertices. The interface vertices are owned by the neighboring submeshes. In the parallel coarsening strategy, we extend these notions of submesh property and submesh neighborhood to the definition of coarse grid levels. The main steps of the parallel coarsening strategy are summarized in the algorithm 2 below.

For a given number of submeshes (i.e. of processes), the coarse grid levels resulting from the application of algorithm 2 are very similar to those obtained in the sequential case. More importantly, they are globally consistent with a sequential agglomeration in the sense that, for a given submesh, the partial agglomeration in the overlapping zone matches perfectly the agglomeration of the neighboring submeshes. This is illustrated in the 2D case on figures 3.2 and 3.3 below.

Algorithm 2 *Parallel grid coarsening by agglomeration.*

FOR *each submesh* DO IN PARALLEL

- *Perform standard sequential agglomeration on the owned vertices (i.e. control volumes)*
- *Send to neighboring submeshes the result of the local agglomeration for interface*
- *Receive the agglomeration*
- *Reproduce the neighboring agglomeration on the not-owned vertices*
- *Construct appropriate communication data structures*

ENDFOR

3.1.5 Numerical and performance results

Calculations have been performed on a SGI Origin 2000 system equipped with Mips R10000/195 Mhz processors. The code is written in Fortran 77 and the Mips F77 compiler has been used with maximal optimization options. The native SGI implementation of MPI has been used. Performance results are given for 64 bit arithmetic computations. In the following tables, N_p gives the number of processors for the parallel execution, N_g is the total number of levels in the multigrid hierarchy (finest mesh included), N_c denotes the number of multigrid cycles used for each linear system solution; "Total time" denotes the total (elapsed) execution time and "CPU time" denotes the total CPU time (taken as the maximum value of the local per process measures). The parallel speedup $S(N_p)$ is always calculated using the elapsed execution times; $G(N_p)$ denotes the overall gain between the multigrid and the monogrid algorithms. Finally, the term "linear threshold" is used to characterize the accuracy of the linear system solves (i.e. the level of reduction of the initial linear residual).

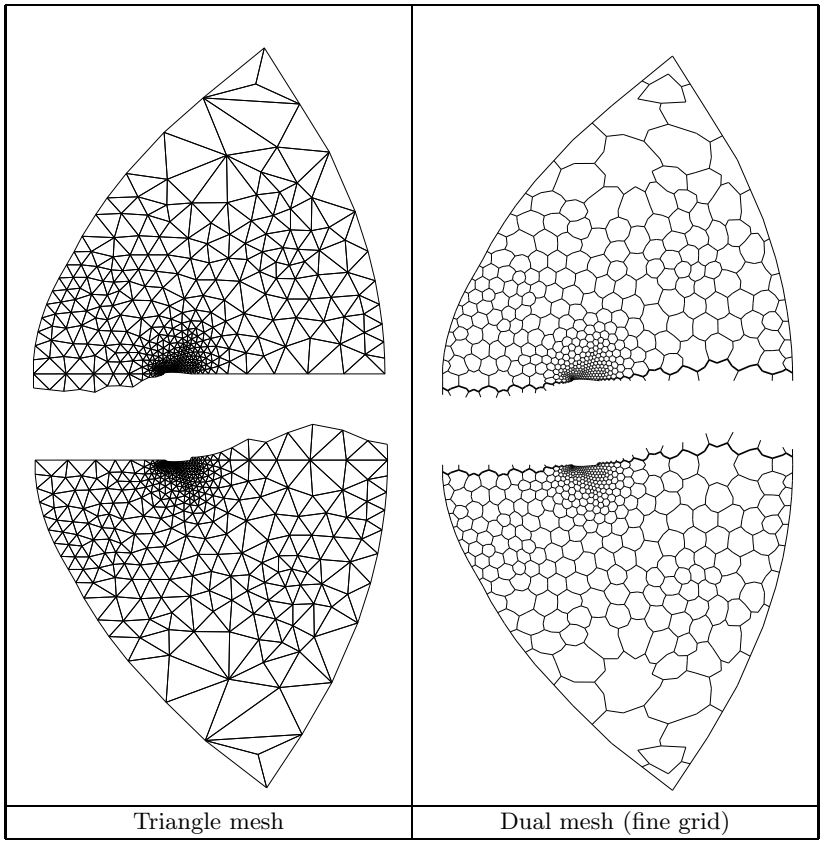
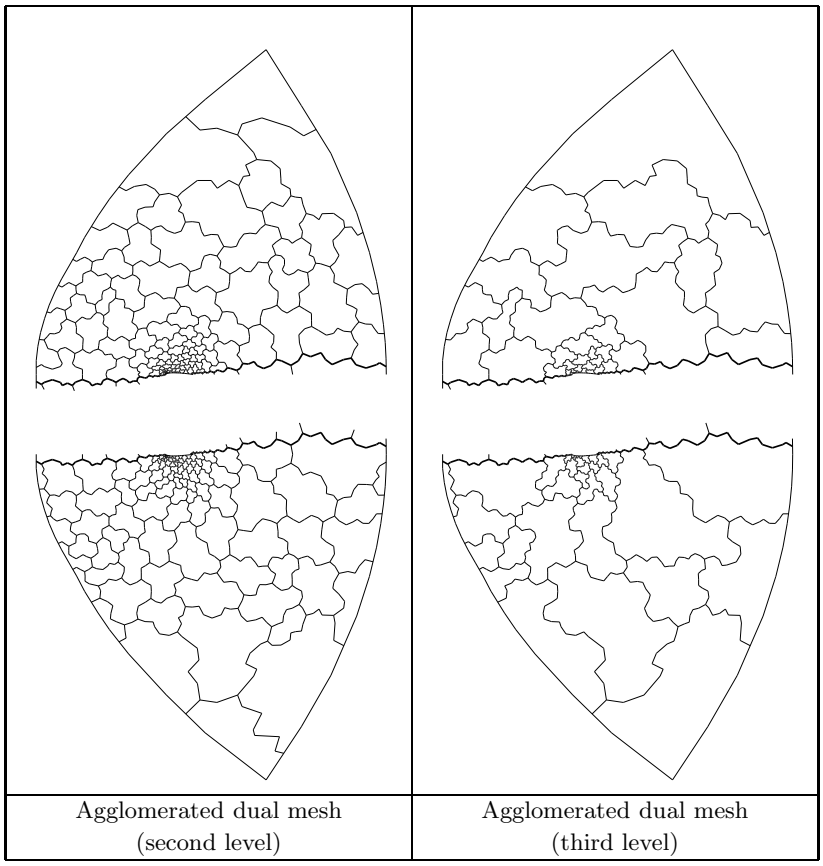


Figure 3.2: Parallel coarsening strategy (first level)



Agglomerated dual mesh (second level)	Agglomerated dual mesh (third level)

Figure 3.3: Parallel coarsening strategy : second and third levels

3.1.5.1 Euler flow around an ONERA M6 wing

We first consider the classical test case given by the inviscid transonic flow around an ONERA M6 wing. The free-stream Mach number is set to 0.84 and the angle of attack to 3.06° (see figure 3.4). Several tetrahedral meshes have been used; their characteristics are summarized in table 3.1 below (meshes M2 and M4 are obtained through a global division of mesh M1). The computation is second order accurate in space. The CFL number has been fixed to 10^6 for all the pseudo-time iterations. The calculation is started from a uniform flow. This test case is mainly used for validation purposes (subsection 3.1.5.1.1) and also to obtain a first estimation of the overall gain when using the multigrid algorithm (subsection 3.1.5.1.2).

Table 3.1: Characteristics of unstructured meshes around an ONERA M6 wing
N_V : # vertices - N_T : # tetrahedra - N_E : # edges

Mesh	N_V	N_T	N_E
M1	2,203	10,053	13,257
M2	15,460	80,424	99,891
M3	31,513	161,830	201,479
M4	115,351	643,392	774,774

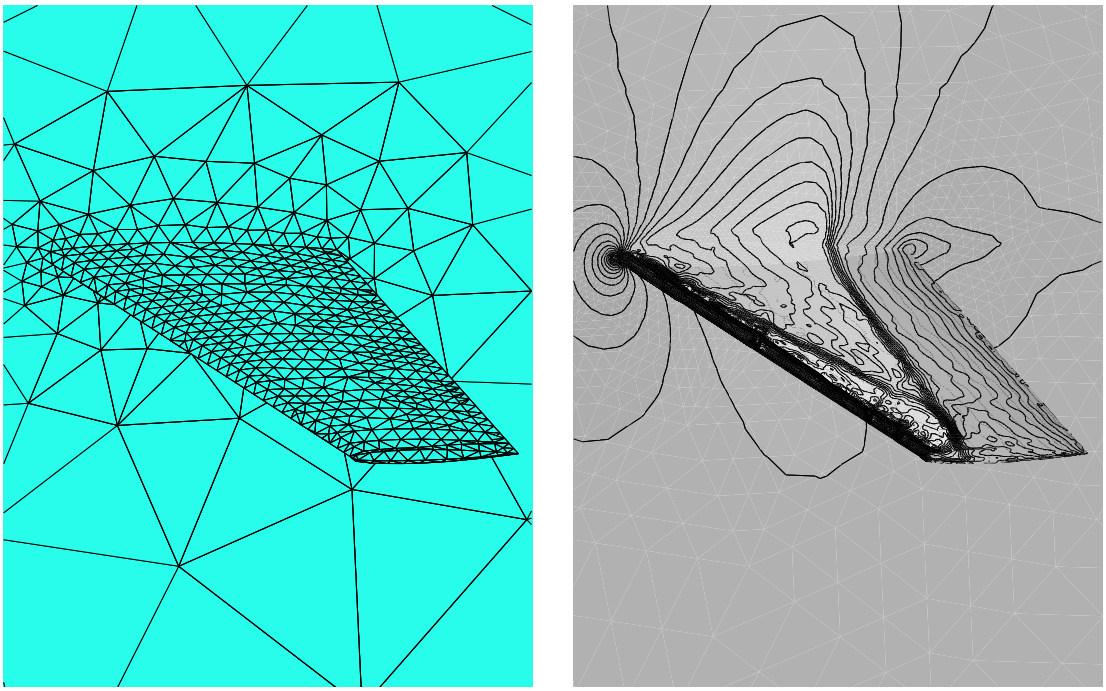


Figure 3.4: Surfacic mesh (mesh M1) and steady contour lines of the Mach number on an ONERA M6 wing

3.1.5.1.1 Assessment of the convergence rate. This series of numerical experiments aims at verifying the most important property of any well designed linear multigrid solution algorithm which states that the rate of convergence is independent of the number of degrees of freedom. In the present context, this property is assessed for the resolution of the linear system resulting from the use of a linearized implicit scheme for time advancing the Euler equations (see section 1.4.1 of chapter 1). For this purpose, we first consider the ideal two-grid algorithm

that is, we construct only one coarse grid level and we perform $\nu_1 = 1$ pre-smoothing step and $\nu_2 = 1$ post-smoothing step while the coarse grid system is solved exactly $\nu_g \to \infty$). Numerical experiments are performed on a single processor of the SGI Origin 2000 system. The smoother is given by the Gauss-Seidel relaxation method. Results are given in table 3.2; the corresponding convergence curves are shown on figure 3.5. In table 3.2, the reduction factor μ_r is defined as :

$$\mu_r = r_{tg}^{\frac{1}{N_c-1}} \quad \text{with} \quad r_{tg} = b_{(1)} - PX_{(1)}$$

where $PX_{(1)} = b_{(1)}$ is the linear system to be solved of the finest grid, N_c denotes the number of ideal two-grid cycles and r_{tg} stands for the residual computed on the finest grid level (the exponent $N_c - 1$ is used because the residual r_{tg} is normalized with the residual computed after the first ideal two-grid cycle). Except for mesh M1 which is too coarse, the required property is exhibited. The same analysis is now performed using the multigrid V-cycle with $\nu_1 = \nu_2 = \nu_g = 1$. Table 3.3 reports the characteristics of the agglomerated coarse grid levels for each of the available meshes. Note that for mesh M3, the last two coarse grid levels should ideally contain 370 and 64 zones; this is not the case here because mesh M3 results from a partial division of mesh M2. Figure 3.6 visualizes the obtained convergence curves.

We conclude that these results validate the developed linear multigrid algorithm, at least for the Euler model which has a purely hyperbolic nature. The same kind of validation has also been performed for the laminar and the turbulent Navier-Stokes equations even though the task is harder in those cases because of the influence of the underlying meshes on the computed solution (e.g. with reference to quantities such as the cell Reynolds number).

Table 3.2: Euler flow around an `ONERA M6 wing`
Ideal two-grid convergence
Numbers of cycles to convergence and associated reduction factors

Mesh	N_c	μ_r
M1	17	0.24
M2	21	0.31
M3	21	0.31
M4	20	0.30

Table 3.3: Euler flow around an `ONERA M6 wing`
Characteristics of the agglomerated coarse grid levels
N_z : numbers of zones (macro control volumes) on each coarse grid level

Mesh	N_g	N_z
M1	3	370/64
M2	4	2203/370/64
M3	4	4341/821/190
M4	5	15460/2203/370/64

3.1.5.1.2 Calculation of the steady flow. We consider now the calculation of the steady flow corresponding to figure 3.4. The objective is to compute the steady state solution of the Euler equations using mesh M4. The total number of unknowns is $115,351 \times 5 = 576,755$ (i.e. 5 degrees of freedom per mesh vertex). When one is concerned with comparing the numerical efficiencies of monogrid and multigrid solution methods such as those considered in this study, a classical difficulty is that the way to proceed is not unique. For instance, a comparison based on the same linear threshold ε for both algorithms is generally unfair for the multigrid algorithm

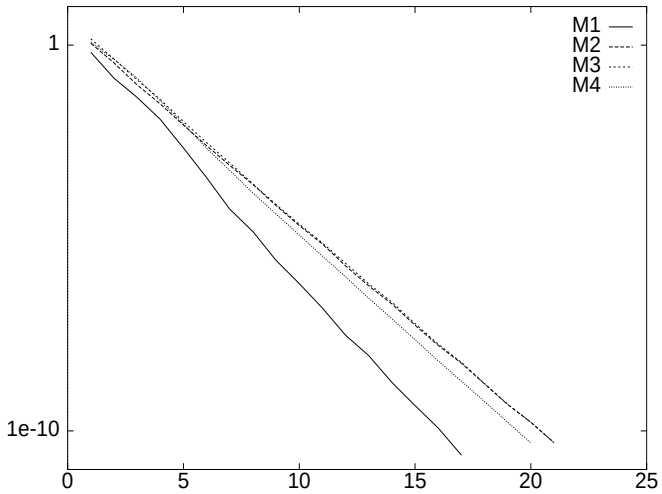


Figure 3.5: Euler flow around an `ONERA M6 wing`
Ideal two-grid convergence
X-axis : number of cycles - Y-axis : residual in log scale

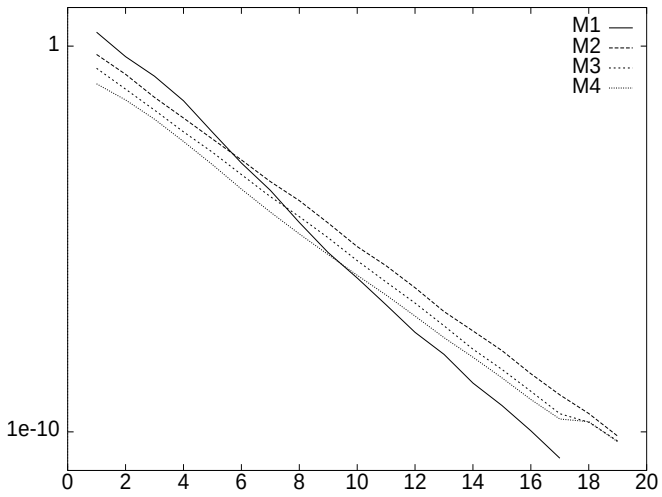


Figure 3.6: Euler flow around an `ONERA M6 wing`
Multigrid V-cycle convergence
X-axis : number of cycles - Y-axis : residual in log scale

because the solution it yields is always of better quality (at least when the underlying linear system solution is only partially converged which is always the case for steady state calculations using the linearized implicit scheme described in section 1.4.1 of chapter 1). The ideal situation is obtained when the computational effort (i.e. the linear complexity) is carefully evaluated for both algorithms through several numerical experiments. Here, several monogrid and multigrid calculations have been performed. For each algorithm, the goal was to find the appropriate number of relaxations (monogrid algorithm) and V-cycles (multigrid algorithm) resulting in the optimal non-linear convergence to steady state. The Jacobi relaxation method has been adopted for both algorithms and the selected V-cycle is characterized by $\nu_1 = 2$ pre-smoothing steps, $\nu_2 = 3$ post-smoothing steps and $\nu_g = 2$ smoothing steps on the coarsest grid level. For the multigrid algorithm, 3 coarse levels have been constructed (thus, the total number of grid levels is $N_g = 4$). Several numerical experiments have been performed and can be summarized as follow:

- for the monogrid algorithm, a constant number of $\nu_f = 75$ Jacobi relaxations at each time step has been found to yield the optimal non-linear convergence to steady state,
- for the multigrid algorithm a constant number of $N_c = 2$ V-cycles is sufficient to demonstrate the same non-linear convergence.

Performance results are given in table 3.4 where "SG" and "MG" respectively denote the monogrid and multigrid algorithms. The super-linear speedup observed on the SGI Origin 2000 system is due to combined memory and cache effects (128 Mb of local memory for a 4 Mb cache size per processor on our system). On 8 processors, the multigrid algorithm is 3 times faster than the monogrid algorithm; this result is rather satisfying given that the flow under consideration is relatively simple and that the equivalent 2D size of the underlying mesh is $115,351^{\frac{2}{3}} \approx 2,370$ vertices which is still very coarse.

Table 3.4: Euler flow around an `ONERA M6` wing, performance results on a SGI Origin 2000 system

N_p	Total time SG	CPU time SG	$S(N_p)$ SG	Total time MG	CPU time MG	$S(N_p)$ MG	$G(N_p)$
2	10260 sec	10210 sec	1.00	3092 sec	3076 sec	1.00	3.3
4	4555 sec	4530 sec	2.25	1586 sec	1578 sec	1.95	3.0
8	2078 sec	2066 sec	5.00	680 sec	675 sec	4.55	3.0

3.1.5.2 Turbulent flow inside an aircraft engine diffuser

The application of the multigrid by volume agglomeration principle to the calculation of complex flows is the main objective of this study. Therefore, we consider here such a situation in the form of the numerical simulation of the turbulent two-component flow inside a geometry of aircraft engine diffuser. At the time of this study, combustion models were not yet implemented in the `N3S-NATUR` monogrid solver; as a consequence, only the convection and the diffusion of chemical species are modeled in the present calculation. The steady state solution of this flow is characterized by a Mach number that ranges between 10^{-4} and 0.3 (see figure 3.7 and 3.8 for views of the steady contour lines of the Mach number and velocity fields in selected cut-planes). The underlying mesh contains $N_V = 149,223$ vertices, $N_T = 797,704$ tetrahedra and $N_E = 977,674$ edges. The total number of unknowns is equal to $149,223 \times 8 = 1,193,784$ (i.e. 8 degrees of freedom per mesh vertex).

At each pseudo-time step, the time increment is computed using a CFL law given by CFL$=\min(50 \times i_t, 20000)$ where i_t denotes the pseudo-time iteration number. In the monogrid algorithm as well as in the multigrid algorithm, the basic relaxation method is an hybrid Gauss-Seidel/Jacobi iteration. To be more precise, since the parallelization of the Gauss-Seidel iteration is not trivial in the context of unstructured meshes, we decide to restrict the application of this relaxation method to purely internal vertices and to resort to the Jacobi method for interface vertices. Strictly speaking, the numerical efficiency of this hybrid method is worse than the one of

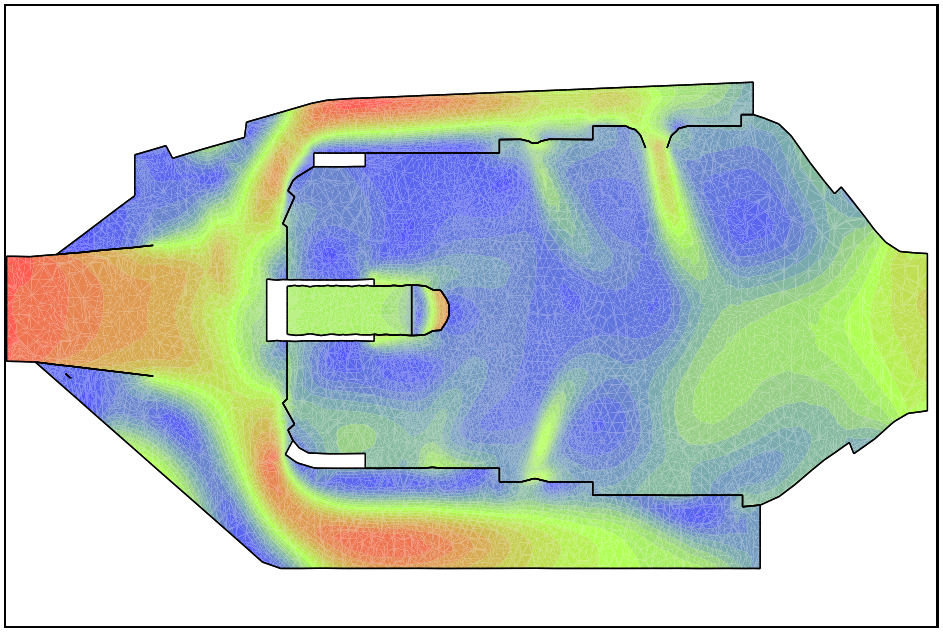


Figure 3.7: Turbulent flow inside an aircraft engine diffuser
Steady contour lines of the Mach number on a selected cut-plane

the classical Gauss-Seidel method, in particular when the number of subdomains is large. The maximum number of relaxations for the monogrid algorithm has been set to $\nu_{f_{max}} = 1500$. For the multigrid algorithm, we have used 3 coarse grid levels (that is $N_g = 4$) and we have selected the following cycles :

- mean flow variables : V-cycle with $\nu_1 = 2$ pre-smoothing steps , $\nu_2 = 4$ post-smoothing steps and $\nu_g = 2$ smoothing steps on the coarsest level;
- turbulent variables and chemical species: V-cycle with $\nu_1 = 2$ pre-smoothing steps , $\nu_2 = 4$ post-smoothing steps and $\nu_g = 2$ smoothing steps on the coarsest level.

For this calculation, the comparison between the monogrid and multigrid methods has been done on a linear threshold basis. For the monogrid algorithm, we have tested two values for the linear threshold at each pseudo-time step : $\varepsilon = 10^{-1}$ and $\varepsilon = 10^{-2}$; for the multigrid algorithm only the value $\varepsilon = 10^{-1}$ has been considered. The obtained non-linear convergences to steady state are shown on figure 3.9 in terms of the evolution of the residual of the density (normalized to its initial value) versus the number of pseudo-time iterations. It is clear from these curves that quasi identical convergences are obtained when $\varepsilon = 10^{-2}$ for the monogrid algorithm and $\varepsilon = 10^{-1}$ for the multigrid algorithm (this behavior illustrates the discussion of the beginning of section 3.1.5.1.2); from now on, the comparison between the two algorithms will be done on the basis of these two choices of the linear threshold. The linear convergences at each pseudo-time step is detailed on figure 3.10 for the systems of mean flow variables and the systems of turbulent variables and for $N_p = 8$ subdomains. The monogrid convergences demonstrate large variations in the number of relaxations. On the contrary, the convergence of the linear systems for the mean flow variables (respectively, for the turbulent variables) requires between 4 to 7 (respectively, between 3 to 5) V-cycles.

Performance results have been obtained on the SGI Origin 2000 system and are given in table 3.5. The overall gains obtained on 8 and 16 processors are clearly seen. From 8 to 32 processors we note a 20% decrease in this gain. There are two main causes to this degradation :

- the cost of the communication steps on $N_p = 32$ subdomains. On the two coarsest grid levels, the ratio of communication to arithmetic operations has a direct impact on the parallel speedup. Moreover, for

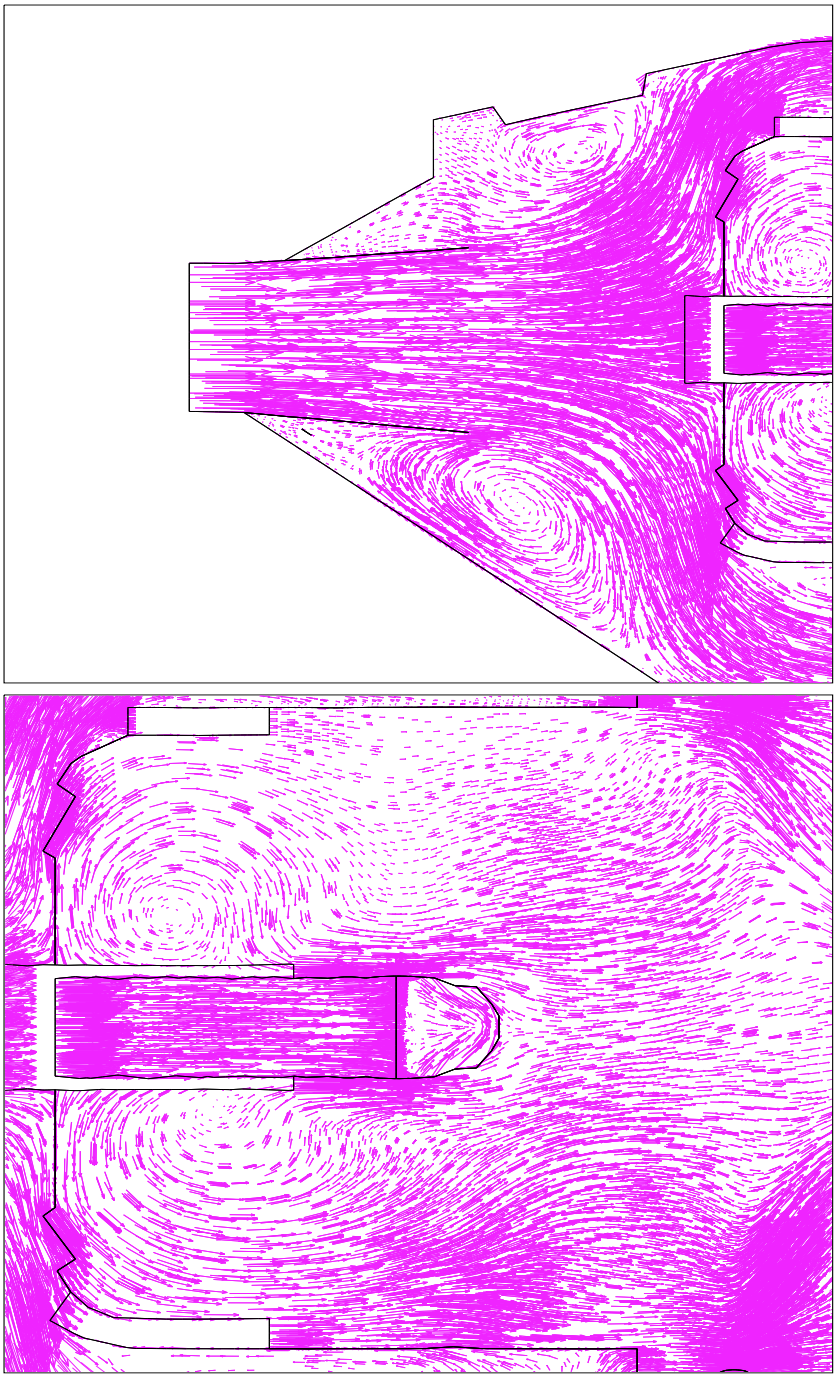


Figure 3.8: Turbulent flow inside an aircraft engine diffuser
Velocity fields in selected cut-planes

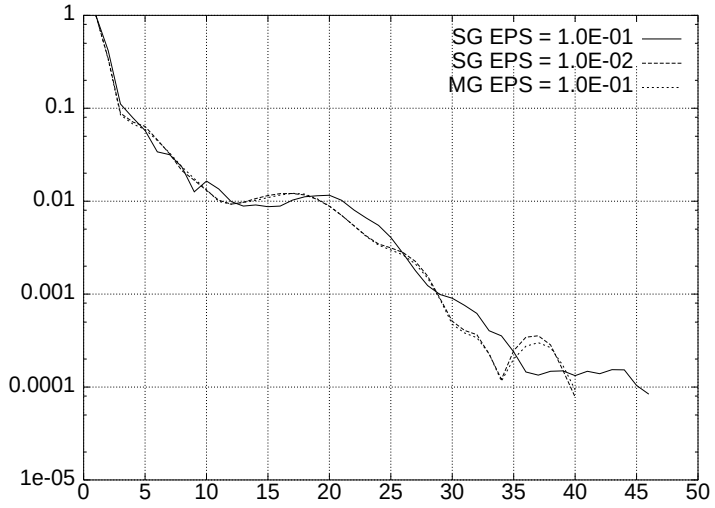


Figure 3.9: Turbulent flow inside an aircraft engine diffuser
Non-linear convergence of the monogrid (SG) and the multigrid (MG) algorithms
X-axis : pseudo-time step - Y-axis : density residual in log scale

each smoothing step on the coarsest grid levels, we need to exchange messages of small sizes, a fact that increases the weight of the message latency in the communication cost;

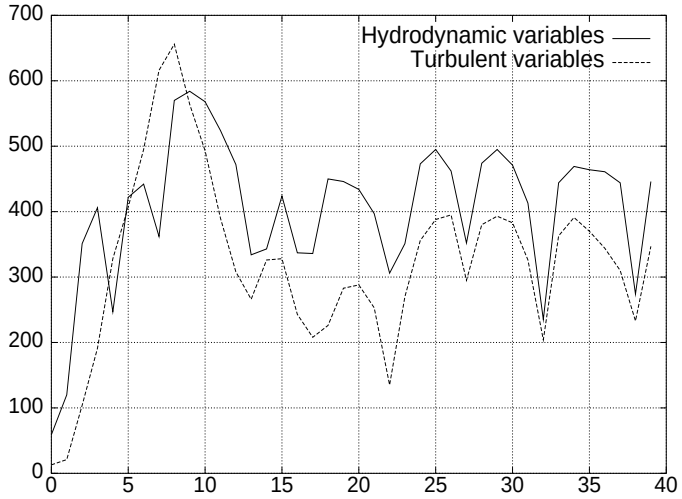
- the overlapping mesh partitioning strategy. As the number of subdomains is increased, the incurred redundant arithmetic operations at the interfaces are such that the equivalent global problem size is larger (between 10% to 15% depending on the number of subdomains) than the size of the original problem. Clearly, this has a direct impact on the overall computational cost.

Table 3.5: Turbulent flow inside an aircraft engine diffuser
Results on a SGI Origin 2000 system

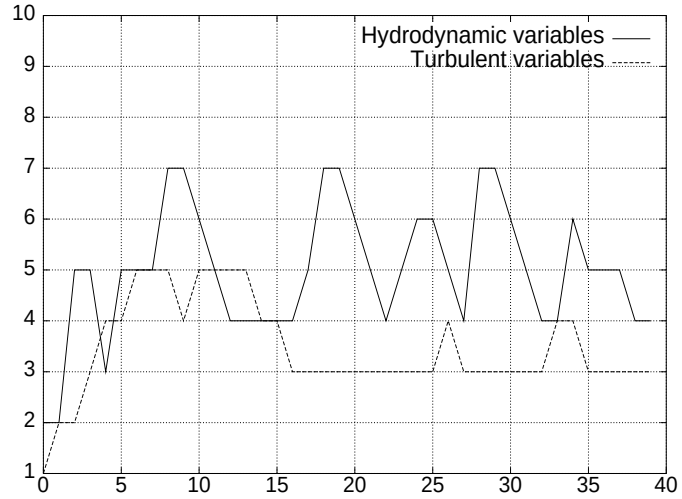
N_p	Total time SG	CPU time SG	$S(N_p)$ SG	Total time MG	CPU time MG	$S(N_p)$ MG	$G(N_p)$
8	16611 sec	16447 sec	1.00	2366 sec	2343 sec	1.00	7.00
16	7356 sec	7284 sec	2.30	1213 sec	1201 sec	1.95	6.00
32	3574 sec	3539 sec	4.65	661 sec	655 sec	3.60	5.40

3.1.5.3 Turbulent flow inside a car engine combustion chamber

Our second example of calculation of complex flows aims at demonstrating the effectiveness of the multigrid principle in situations where the basic relaxation methods such as Jacobi or Gauss-Seidel are unable to solve the underlying algebraic systems due to an increased stiffness of the associated Jacobian matrix. In the present case, the bad conditioning of the algebraic systems obtained at each pseudo-time step results from combined



Monogrid linear convergence ($\varepsilon = 10^{-2}$)
X-axis : pseudo-time step - Y-axis : effective number of relaxations



Multigrid linear convergence ($\varepsilon = 10^{-1}$)
X-axis : pseudo-time step - Y-axis : effective number of cycles

Figure 3.10: Turbulent flow inside an aircraft engine diffuser
Linear convergence of the monogrid (SG) and the multigrid (MG) algorithms ($N_p = 8$)

physical and numerical aspects. The problem under consideration is concerned with the calculation of the steady turbulent flow inside a car engine combustion chamber. As with the previous test case, combustion effects are not modeled. The underlying flow is highly subsonic (the Mach number obtained at steady state ranges from 10^{-4} to 10^{-1}) which is by itself a first source of physical stiffness. In order to compute this subsonic flow accurately, a preconditioned Roe-Turkel variant of Roe's Riemann solver[36] is used for the computation of the convective fluxes. This numerical features is the second source of increased stiffness of the linear systems to be solved at each pseudo-time iteration. The computational mesh contains $N_V = 332,818$ vertices, $N_T = 1,903,704$ tetrahedra and $N_E = 2,267,209$ edges. The total number of unknowns is equal to $332,818 \times 7 = 2,329,726$ (i.e. 7 degrees of freedom per mesh vertex). Figure 3.11 compares the velocity field in a selected cut-plane resulting from the classical Roe scheme and the preconditioned Roe-Turkel variant. These figures clearly show the superiority of the latter scheme in terms of numerical diffusion.

At each pseudo-time step, the time increment is computed using a CFL law given by CFL=$\min(5 \times i_t, 20000)$ where i_t denotes the pseudo-time iteration number (with $1 \leq i_t \leq i_{t_{max}}$). In the monogrid algorithm as well as in the multigrid algorithm, the basic relaxation method is again given by the hybrid Gauss-Seidel/Jacobi iteration. For both algorithms, the linear threshold has been fixed to $\varepsilon = 10^{-2}$. The maximum number of relaxations for the monogrid algorithm has been set to $\nu_{f_{max}} = 1500$. Here, we are interested in comparing the efficiency of various multigrid cycles. This comparison will be limited to the solution of the linear systems for the mean flow variables (i.e. the multigrid cycle used for the solution of the linear systems for the turbulent variables is kept unchanged). For the multigrid method, 3 coarse grid levels have been used (that is $N_g = 4$) in combination with the following cycles :

- for the mean flow variables three cycling strategies have been tested :
  - V-cycle with $\nu_1 = 4$ pre-smoothing steps, $\nu_2 = 6$ post-smoothing steps and $\nu_g = 4$ smoothing steps on the coarsest level,
  - F-cycle with $\nu_1 = 2$ pre-smoothing steps, $\nu_2 = 4$ post-smoothing steps and $\nu_g = 2$ smoothing steps on the coarsest level,
  - W-cycle with $\nu_1 = 2$ pre-smoothing steps, $\nu_2 = 4$ post-smoothing steps and $\nu_g = 2$ smoothing steps on the coarsest level;
- turbulent variables : V-cycle with $\nu_1 = 1$ pre-smoothing steps , $\nu_2 = 1$ post-smoothing steps and $\nu_g = 1$ smoothing steps on the coarsest level.

The obtained non-linear convergences to steady state are shown on figure 3.12 in terms of the evolution of the residual of the density (normalized to its initial value) versus the number of pseudo-time iterations. The effective numbers of iterations of the monogrid and multigrid solution methods that are performed at each pseudo-time step are detailed on figure 3.13. Performance results have been obtained on 12 and 24 processors of the SGI Origin 2000 system and are summarized in table 3.6. The monogrid algorithm is clearly not a reasonable choice for this application because of the important computational effort induced by the solution of the linear systems for the mean flow variables. Most of this effort is wasted in trying to solve the low frequency components of the error which are here the dominant ones due to the subsonic feature of the underlying flow. The multigrid V-cycle is seen to bring an appreciable reduction in the simulation time, however this reduction is not as much important as one could expect. Increasing the number of pre- and post-smoothing steps would probably improve the situation in terms of the required number of V-cycles at each pseudo-time step but at the expense of an increased execution time. The current values of these parameters are already high and increasing them further will yield a V-cycle no more representative of usual cycle configurations. A more elegant way to eradicate the problem would consist in adopting a more efficient smoother. More importantly, the F-cycle and W-cycle are very efficient in the present context. The execution time on 12 processors of the SGI Origin 2000 system is reduced from 14 hours to one hour for both algorithms with a slight advantage for the W-cycle.

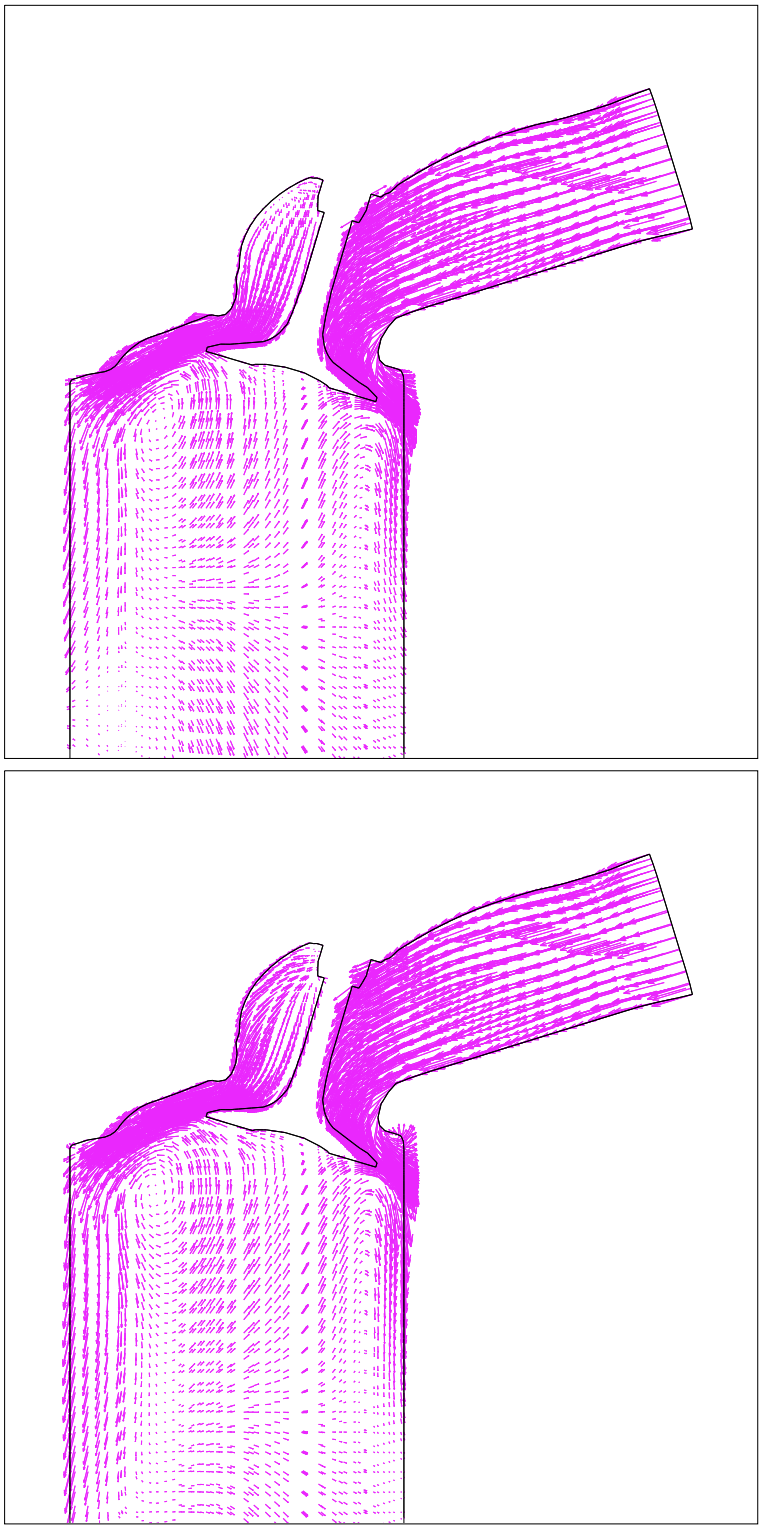


Figure 3.11: Turbulent flow inside a car engine combustion chamber
Steady contour lines of the Mach number in a selected cut-plane
Top figure: Roe scheme - Bottom figure: Roe-Turkel scheme

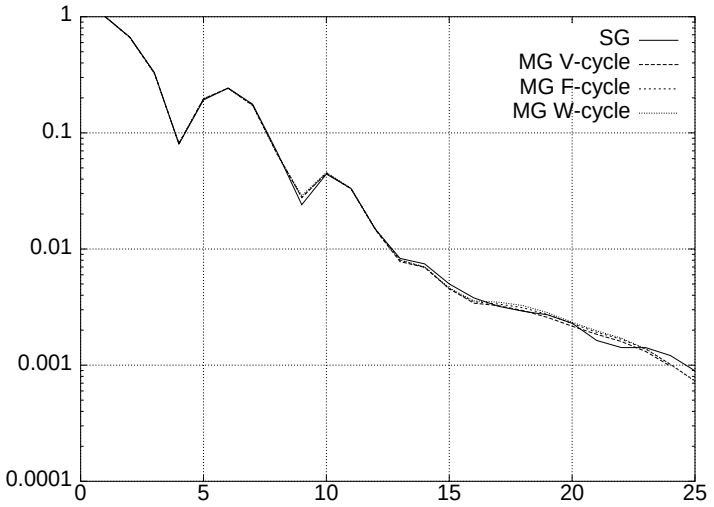
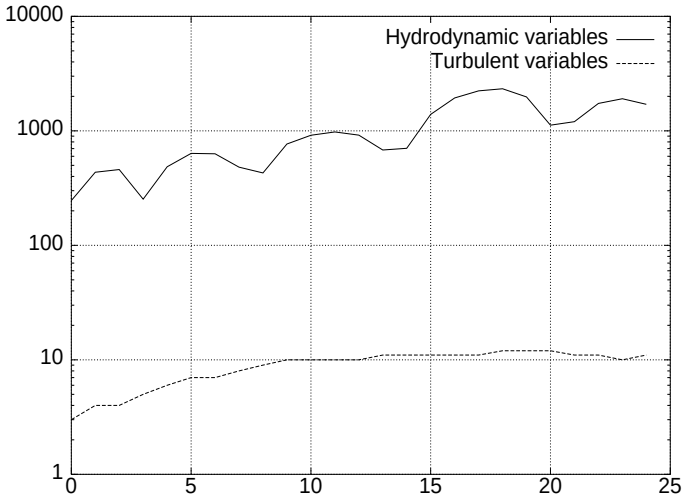


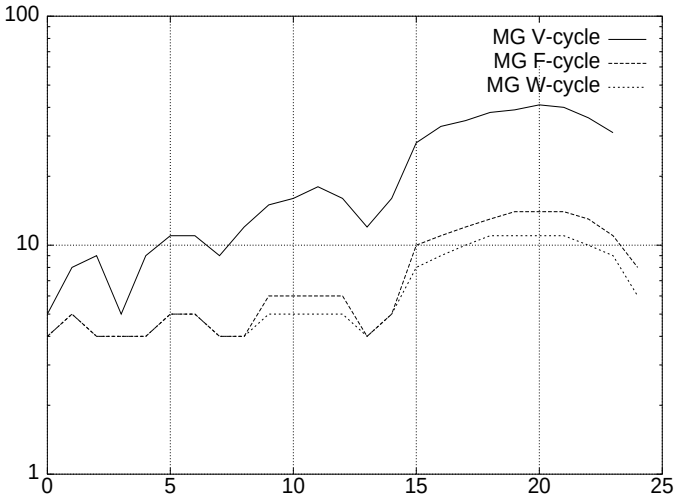
Figure 3.12: Turbulent flow inside a car engine combustion chamber
Non-linear convergence of the monogrid and the multigrid algorithms
X-axis : pseudo-time step - Y-axis : density residual in log scale

Table 3.6: Turbulent flow inside a car engine combustion chamber
Performance results on a SGI Origin 2000 system

N_p	ALGO	Total time	CPU time	$S(N_p)$	$G(N_p)$
12	SG	14 h 06 mn	14 h 03 mn	1.0	-
24	SG	6 h 46 mn	6 h 44 mn	2.1	-
12	MG (V-cycle)	3 h 20 mn	3 h 18 mn	1.0	4.2
24	MG (V-cycle)	1 h 34 mn	1 h 33 mn	2.1	4.3
12	MG (F-cycle)	1 h 15 mn	1 h 13 mn	1.0	11.3
24	MG (F-cycle)	39 mn	38 mn	2.0	10.4
12	MG (W-cycle)	1 h 05 mn	1 h 04 mn	1.0	13.0
24	MG (W-cycle)	35 mn	34 mn	1.9	11.6



Monogrid linear convergence ($\varepsilon = 10^{-2}$)
X-axis : pseudo-time step - Y-axis : effective number of relaxations



Multigrid linear convergence (system of hydrodynamic variables only, $\varepsilon = 10^{-2}$)
X-axis : peudo-time step - Y-axis : effective number of V-cycles

Figure 3.13: Turbulent flow inside a car engine combustion chamber
Linear convergence of the monogrid (SG) and the multigrid (MG) algorithms ($N_p = 12$)

3.1.6 Directional coarsening strategy

The application of a multigrid by volume agglomeration method to the calculation of flows on anisotropic meshes is a non-trivial task which is the subject of several ongoing efforts in the multigrid community. For CFD applications, relevant practical situations are for example given by the calculation of steady turbulent flows on highly stretched meshes[57]-[93] and the calculation of unsteady flows in deforming geometries[109]. In these contexts, the underlying discretization grids often exhibit elements with high aspect ratio, a characteristic that results in serious convergence problems. As a matter of fact, the efficiency of the standard agglomerated multigrid method (i.e. based on an isotropic coarsening strategy) degrades and, in some cases, the solution procedure may fail to converge. This deficiency is even more acute when the adopted smoother is a node-wise relaxation method as it is the case in the present study. A simple explanation of this behavior of the multigrid algorithm is the following: when full coarsening is used (i.e. isotropic coarsening), the iterative process used as a smoother only damps the high frequencies associated to the space direction which is orthogonal to the mesh stretching. Two main strategies can be adopted in order to remedy to this problem: either the iterative process is adapted to efficiently damp the high frequencies aligned with the direction of mesh stretching (e.g. by resorting to directional relaxation or even ILU smoothing[145]) or, the coarsening algorithm is modified to allow for an efficient treatment of the high frequency components on a coarse grid that have not been damped on the finest grid. In this section, we propose a directional coarsening strategy that takes into account a particular metric in the agglomeration procedure used for the construction of the coarse grid levels. Note that the standard coarsening algorithm given in section3.1.3.2 has a purely topological nature. In particular, the resulting agglomeration patterns are tightly linked to the numbering of the vertices in the original mesh. Here, we use a different criterion for the construction of the macro control volumes on the coarse grid levels. This criterion is based on geometrical features of the grid level to be agglomerated and essentially aims at taking into account a local direction of mesh stretching. To summarize, the new coarsening strategy combines two mechanisms: on one hand, local metrics i.e. stretching directions and strength are identified and, on the other hand, the standard agglomeration procedure is adapted to take into account these local metrics.

The definition of a directional coarsening criterion generally relies on the determination of strong connections. This can be done in several ways. In the two-dimensional case, the approach proposed in [57] is inspired by an idea used in algebraic multigrid: the strong connections are evaluated by measuring the coefficients of the finite element Laplace operator. Then, the values of these coefficients are used to compute a weight for each connection between two macro control volumes. This strategy has been extended to the three-dimensional case and is briefly described below.

Let A_{ij}^k $(\forall (i,j) \in U^{k-1})$ denote the coefficients of the discrete Laplace operator A^k defined on grid level G_k. These coefficients are given by :

$$A_{ij}^k = \iiint \overrightarrow{\nabla} \varphi_i^k . \overrightarrow{\nabla} \varphi_j^k \; d\vec{x} \;\; \text{with} \;\; \varphi_j^k = \sum_{p \in U^{k-1}(j)} \varphi_p^{k-1}$$

where $U^{k-1}(j),\ j = 1, ..., N_k$ stands for the set of indices of the macro control volumes C_l^{k-1} of grid level G_{k-1} defining the macro control volume C_j^k of grid level G_k. The connection between C_i^k and C_j^k is denoted by $< i, j >$. Then, we consider that $< i, j >$ defines a strong connection if :

$$|A_{ij}^k| \geq \epsilon \max_{p \in N_i^k} |A_{ip}^k| \quad \text{with} \quad \epsilon = \frac{1}{4}$$

In the above expression, N_i^k denotes the set of indices of the macro control volumes that are neighbors of C_i^k. The choice $\epsilon = \dfrac{1}{4}$ is rather classical and means that the directional coarsening is applied when the cell aspect ratio is either lesser than $\dfrac{1}{2}$ or greater than 2.

Let S_i^k denote the set of $j \in N_i^k$ such that $< i, j >$ defines a strong connection. For each macro control volume C_i^k, we denote by j_0 the index such that $< i, j_0 >$ is the strongest connection :

$$A^k_{ij_0} = \max_{j \in S^k_i} |A^k_{ij}|$$

Let g^k_i be the mass center of the macro control volume C^k_i of grid level G_k. Then, one can evaluate the following vectors:

$$\vec{e^k_\xi} = \frac{\overrightarrow{g^k_i g^k_{j_0}}}{\|\overrightarrow{g^k_i g^k_{j_0}}\|} \quad \text{and} \quad \vec{e_\eta} \quad \text{such that} \quad \vec{e^k_\eta} \perp \vec{e^k_\xi}$$

and a stretching intensity factor can be defined as:

$$L^k_i = \frac{\sum_{j \in S^k_i} |\overrightarrow{g^k_i g^k_j}.\vec{e^k_\xi}|}{\sum_{j \in S^k_i} |\overrightarrow{g^k_i g^k_j}.\vec{e^k_\eta}|}$$

In the two-dimensional case, the directional agglomeration will rely on the determination of the unique $C^k_{j_0}$ associated to the strongest connection. In the three-dimensional case, one has to select several such indices because there is generally more than one direction of stretching.

The directional coarsening algorithm is given below.

`Algorithm 3` *Directional grid coarsening algorithm by agglomeration in the 2D case for the construction of the coarse grid level G_{k+1} from the fine grid level G_k*

`[0]` *Set $j = 0$ (j stands for the counter of coarse grid control volumes)*

`[1] LOOP` *on each control volume C^k_i of grid level G_k*

`[2] IF` C^k_i has already been included in a group $C^{k+1}_q \quad q = 1, \cdots, j-1$ `THEN`

`[2a]` Consider the next control volume i.e. `GOTO [1]`

`ELSE`

`[2b]` Set $j \leftarrow j+1$ and create a new group C^{k+1}_j containing C^k_i

`[2c]` Identify the index j_0 such that $< i, j_0 >$ is the strongest connection

`[2d]` Compute the stretching intensity factor L^k_i

`[3] IF` $L^k_i \in [0.5, 2]$ `THEN`

`[3a]` Perform isotropic agglomeration:

$$\boxed{C^{k+1}_j = C^k_i \bigcup_{p \in N^k_i} C^k_p \quad \text{such that} \quad C^k_p \notin C^{k+1}_q, \quad q = 1, \cdots, j-1}$$

`[3b] GOTO [1]`

`ELSE`

`[3c]` Perform directional agglomeration along the strongest connection:

$$\boxed{C^{k+1}_j = C^k_i \bigcup C^k_{j_0}}$$

`[3d] GOTO [1]`

`[3 end] ENDIF`

`[2 end] ENDIF`

[1 end] ENDLOOP

The algorithm used in the three-dimensional case is essentially the same except for step [3c] where the directional agglomeration is performed along α directions instead of one.

The directional agglomeration strategy is illustrated on figures 3.14 to 3.17 for a model 2D geometry of a piston engine chamber.

The application of the agglomerated multigrid method based on the proposed directional coarsening strategy to turbulent flows on highly stretched meshes or to unsteady flows in deforming geometries is the object of an ongoing effort. However, in order to have a first idea of the potential improvements of the multigrid efficiency, we present here a preliminary comparison between the standard and directional coarsening strategies using the test case considered in section 3.1.5.2 (as a matter of fact, the underlying mesh is moderately stretched). Calculations have been performed for three values of the linear threshold: $\varepsilon \in \{10^{-2}, 10^{-3}, 10^{-4}\}$. Timing results are given in table 3.7 for the computation of the steady state in the same conditions (multigrid cycles definition, non-linear threshold) than those considered in section 3.1.5.2, on 8 processors of the SGI Origin 2000 system. For this particular geometry and associated tetrahedral mesh, the memory overhead between the single grid solver and the multigrid solver based on the standard coarsening strategy is 30%; using the directional coarsening strategy, this overhead raises to 37.5%. As can be expected, the main consequence of this fact is that a single V-cycle is more costly in the latter case because on each coarse grid level there are more unknowns due to a lower coarsening ratio. Indeed, for $\varepsilon \in 10^{-1}$ we did not observe any notable gain in overall CPU time and this is the reason why this situation is not reported here. In table 3.7, $\mathrm{R}_{stand}^{dirc} = (\text{Total time}_{stand} - \text{Total time}_{dirc}) / \text{Total time}_{stand}$.

In order to have a more precise idea of the efficiency of the two multigrid algorithms, we visualize on figure 3.18 the linear convergences at each time step for the system of mean flow variables and for $\varepsilon = 10^{-4}$. The multigrid algorithm based on the directional coarsening strategy requires almost the same number of V-cycles at each time step; for most of the computation this number is approximately half of what is required by the multigrid algorithm based on the standard agglomeration strategy. The aggregate numbers of V-cycles for both algorithms are respectively equal to 1016 and 619 i.e. a reduction of 39% is obtained with the multigrid algorithm based on the directional coarsening strategy. In terms of CPU times, the gain is less important (27%) because of the reason mentioned in the previous paragraph.

Table 3.7: Turbulent flow inside an aircraft engine diffuser
Comparison between standard and directional coarsening algorithms
Results on a SGI Origin 2000 system ($N_p = 8$)

ε	Total time MG-stand	CPU time MG-stand	Total time MG-dirc	CPU time MG-dirc	$\mathrm{R}_{stand}^{dirc}$
10^{-2}	3229 sec	3197 sec	2856 sec	2827 sec	11.5%
10^{-3}	4796 sec	4748 sec	4060 sec	4019 sec	15.5%
10^{-4}	7698 sec	7622 sec	5605 sec	5550 sec	27.0%

3.1.7 Conclusion

Parallel computing offers the opportunity to simulate compressible flows of increasing physical complexity, around or within complex geometries in reasonable amounts of time. However, designing robust and efficient solvers on unstructured finite element meshes remains a challenging objective that should not be neglected. It is now widely accepted that the successful application of simple solution methods exhibiting high levels of parallel efficiency is often limited to simple problems. As a general rule, the ideal solver should offer a good compromise between parallel and numerical efficiency. An example of such an approach has been considered here through the development of parallel multigrid algorithms to accelerate the various linear system solution steps of an industrial flow solver. The resulting parallel multigrid flow solver is built around two main components:

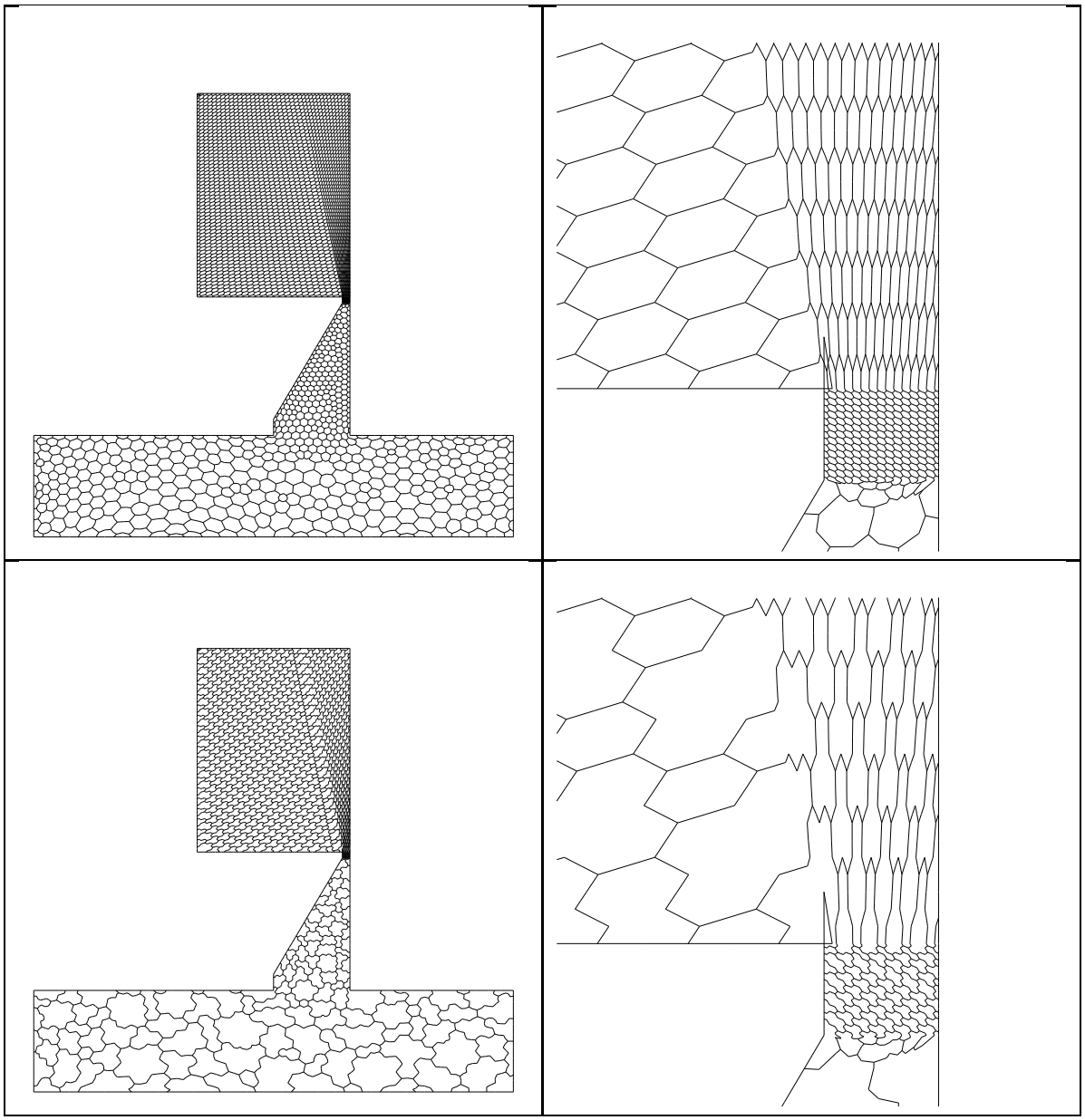


Figure 3.14: Isotropic agglomeration strategy
Fine grid level and first coarse grid levels

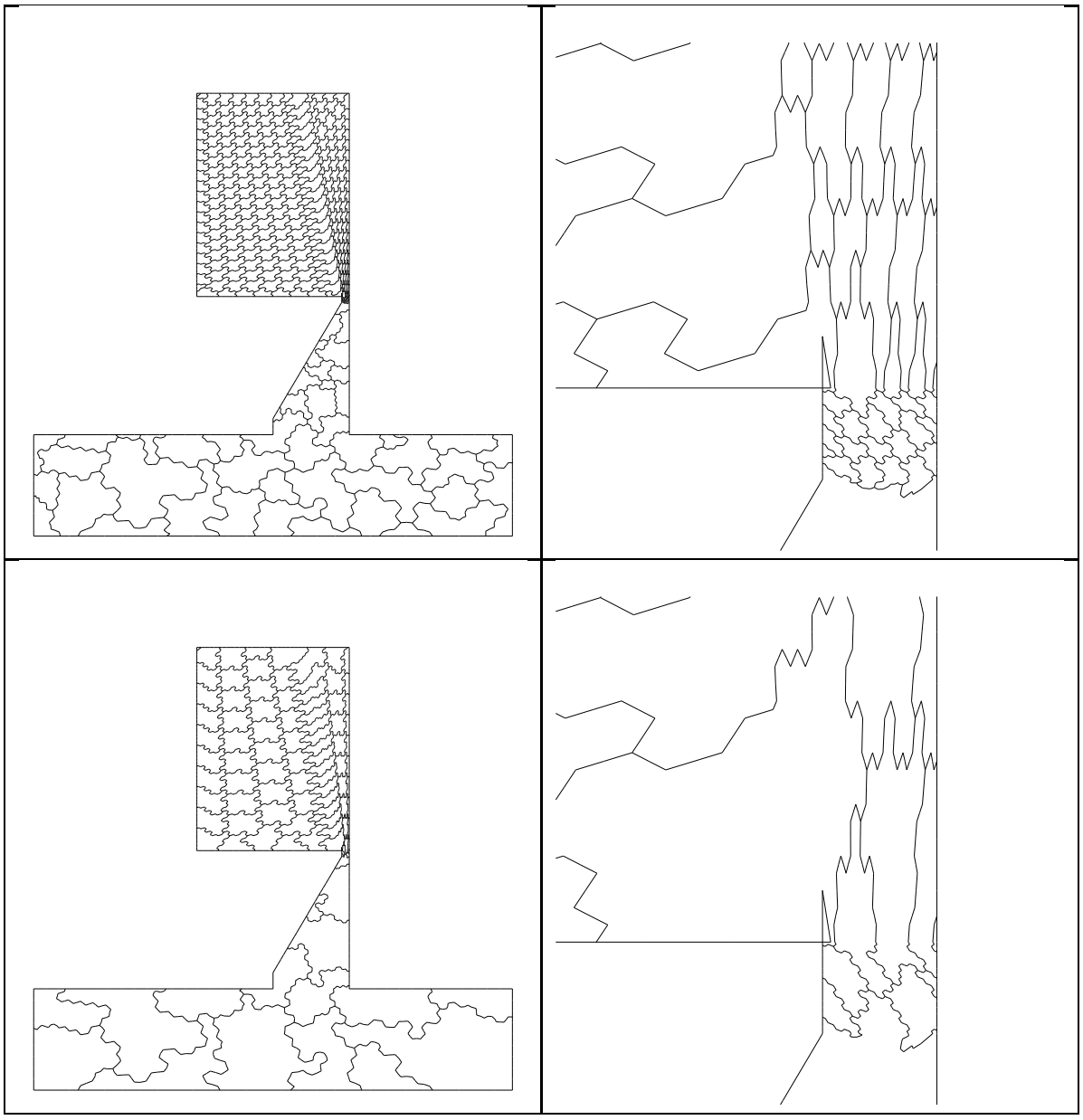


Figure 3.15: Isotropic agglomeration strategy
Second and third coarse grid levels

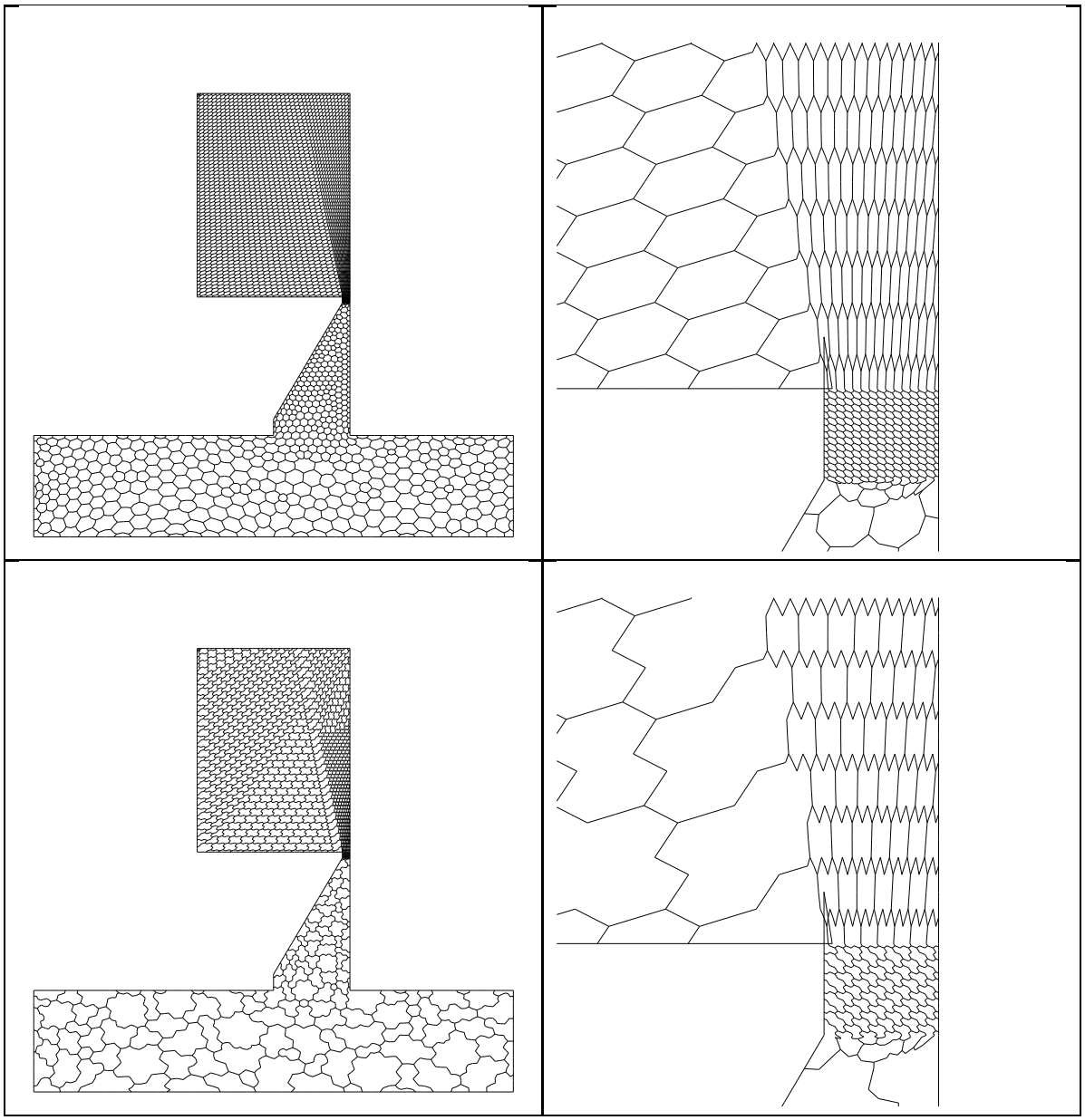


Figure 3.16: Directional agglomeration strategy
Fine grid level and first coarse grid levels

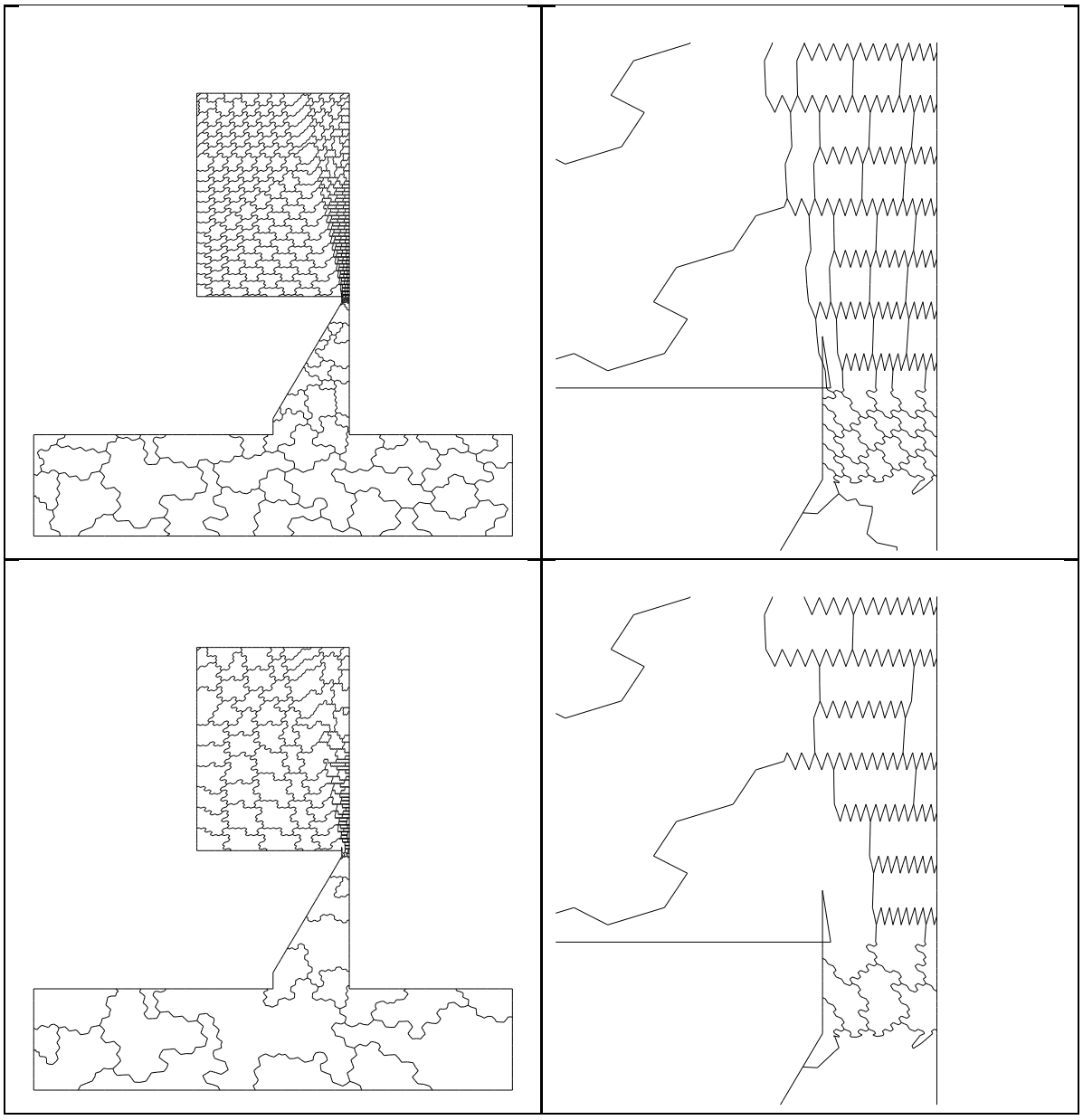
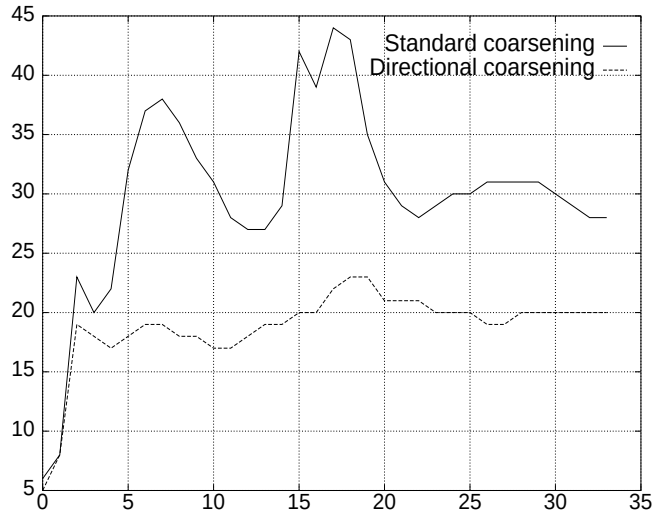


Figure 3.17: Directional agglomeration strategy
Second and third coarse grid levels



X-axis : pseudo-time step - Y-axis : effective number of cycles

Figure 3.18: Turbulent flow inside an aircraft engine diffuser
Linear convergence of the the multigrid algorithm ($N_p = 8$, $\varepsilon = 10^{-4}$)
Comparison of standard and directional coarsening strategies

- a widely adopted strategy for the SPMD parallelization of finite element type calculations. This strategy maximizes the parallel efficiency of the resulting solver by explicitly enforcing data locality through domain partitioning techniques (see for example [84]);
- a linear multigrid acceleration technique for the solution of large sparse linear systems arising from the adoption of a linearized implicit time integration technique for time advancing the semi-discrete equations. With respect to the mixed finite element/finite volume formulation used for spatial discretization in the monogrid solver, a multigrid by volume agglomeration strategy has been selected. Its main advantage relies in the fact that the multigrid hierarchy can be automatically generated using the sole data given by the finest discretization of the computational domain. This aspect is of particular importance in the context of the SPMD parallelization strategy considered in this study: the problem of generating local data structures for coarse grid topologies and data exchange at submesh interfaces is treated in parallel without resorting to an appropriate (multi-mesh) partitioning technique.

The standard agglomerated multigrid strategy relies on an isotropic coarsening algorithm for the construction of the coarse discretizations. Using such a strategy, appreciable overall gains have been demonstrated here for steady flow calculations of industrial relevance, using tetrahedral discretizations containing from 100,000 to 330,000 mesh vertices. Larger gains are expected for unsteady flows inside deforming geometries (e.g. piston engine flows) for which the linear systems need to be solved more accurately, therefore requiring much more relaxations in the single grid case. However, such calculations are also characterized by highly stretched meshes that induce a notable degradation of the multigrid efficiency when using the isotropic coarsening strategy. In preparation of the treatment of this class of problems, we have proposed a directional coarsening algorithm which is based on an appropriate definition of strong connections between macro control volumes. A preliminary evaluation of the resulting directional agglomerated multigrid algorithm has been performed on a steady turbulent flow calculation using a moderately stretched mesh, and has demonstrated the merits of the approach.

3.2 Multiplicative and additive multigrid algorithms

3.2.1 Introduction

Thanks to the numerical results given in the previous section, we have demonstrated that the multigrid by volume agglomeration principle can yield an efficient solver for the calculation of complex compressible flows on unstructured meshes. However, these results have also shown that the parallel efficiency of the multigrid solver degrades more notably than that of the monogrid solver. This unfavorable behavior of the multigrid solver is classically explained as follows: when adopting a standard parallelization strategy that relies on domain partitioning for the parallel treatment of the pre- and post-smoothing steps, whereas the coarse grid levels are visited sequentially according to predefined cycles (V-cycle, F-cycle or W-cycle), the main drawback is that as the calculation in a given cycle proceeds from the finest level to the coarsest ones the ratio between communication and calculation becomes worse resulting in the observed degradation of the parallel efficiency. Such an approach can be characterized as an $intra\text{-}level$ parallelization strategy. Moreover, the intrinsic sequential treatment of the coarse grid levels in the V-, F- and W-cycles is dictated by the $multiplicative$ nature of the standard multigrid algorithm. Clearly, reducing communication overheads is of crucial importance for a wider adoption of parallel multigrid methods. One possible direction of research aims at designing $additive$ multigrid formulations similarly to what is done for domain decomposition methods[130].

In this section, we study in details the properties of two multigrid formulations, both from the numerical and parallel efficiency viewpoints. This is first done by conducting an abstract convergence analysis using a model problem. Then, we compare the two methods through numerical experiments. The adopted numerical framework is consistent with the one adopted in section 3.1 however, in this study, we focus our attention on the two-dimensional case and more particularly on the numerical simulation of inviscid flows. The starting point of our study is given by a monogrid solver for the Euler equations. The spatial discretization combines finite element and finite volume concepts and is designed on unstructured triangular meshes. Steady state solutions of the resulting semi-discrete equations are obtained by using a linearized Euler implicit time advancing strategy. Then, each pseudo-time step requires the solution of a sparse linear system for the flow variables. In the monogrid solver this system is approximately solved using several sweeps of a standard relaxation method (the Jacobi or the Gauss-Seidel method). Moreover, the monogrid solver is parallelized using a domain partitioning approach and a message passing programming model. In this context, the present work aims at examining how parallel multigrid acceleration can be used to improve the efficiency of the existing linear system solution strategy. As in section 3.1, a linear multigrid by volume agglomeration method is the basis of this study. As mentioned above, two parallel multigrid formulations are considered here. The first formulation is the most standard one that extends the domain partitioning approach to the treatment of the smoothing steps on each coarse grid level (i.e. the parallelization strategy actually adopted in section 3.1). The second formulation is based on a residual/correction filtering approach inspired from the work of Chan and Tuminaro[30]. We note in passing that this approach is different from the one studied by Bastian $et\ al.$[10]. The resulting parallel multigrid algorithm theoretically allows the simultaneous treatment of all the grid levels ($inter\text{-}level$ parallelism).

The literature on parallel multigrid methods is relatively rich. For a broad overview of parallel multigrid algorithms, we refer the interested reader to the recent survey compiled by Jones and McCormick[79]; see also the paper of Douglas[43]. Parallel multigrid algorithms can be classified into three families.

A first family gathers the so-called non-telescoping multigrid algorithms. Multiple-coarse grid methods[29]-[58] and concurrent (or subspace-parallel) methods[59] belong to this class; the method of Chan and Tuminaro[30] considered here is a subspace-parallel method.

The second family includes methods that are based on the standard multigrid algorithms associated to some techniques to minimize inter-processor communications and to let the idle processors work on something else[91]-[16]. The method of Brandt and Diskin[16] is somewhat unusual as it is designed to reduce the need for communications on the finest levels of a multigrid hierarchy. The basic idea is the one that has inspired segmental-refinement-type procedures which were proposed to overcome storage problems on sequential computers. The starting point is a re-interpretation of the non-linear FAS (Full Approximation Storage) method. In the FAS method, the calculation on the fine grid level can be viewed as a correction to the coarse grid problem; this

correction is in fact a *defect correction* that ensures that the coarse grid solution has the same accuracy as the fine grid one. The strategy consists in introducing some approximation in the calculation of this defect correction by resorting to local calculations on segments of a fine grid level instead of computing a globally consistent value. In the context of a parallel implementation of the algorithm, each segment is treated by a separate process in a domain partitioning approach and the defect correction is approximated by neglecting the communication operations in the pre- and post-smoothing steps.

Finally, the third class of algorithms aims at selecting a proper finer grid as the new coarsest grid such that the number of unknowns on all grids is greater than the number of processors therefore avoiding the idle processor problem. The U-cycle proposed by Xie and Scott[147] is a recent example of a multigrid algorithm of this class. In [147], a detailed analysis reveals that the U-cycle with a finer coarsest grid can have a faster convergence rate, and the coarsest grid equations of the U-cycle can be solved approximately without increasing the total number of cycles over what would be required using exact coarsest grid solutions. A parallel U-cycle is simply defined using a domain partitioning technique. A time complexity analysis demonstrates that the parallel U-cycle is fully scalable and can have a super-linear speed-up in comparison to the original V-cycle. Application of the method to a five-point finite difference discretization of a model Poisson problem on a unit square and experiments on an Intel Paragon and an IBM SP2, tend to confirm the theory.

From an experimental standpoint, most of the algorithms referenced previously have only been applied to model (elliptic) problems using 2D or 3D uniform (finite difference) discretizations of simple computational domains. As mentioned in the conclusion of [79], the development and application of non-standard parallel multigrid methods to real world problems is still in its infancy. A preliminary study of the parallel multigrid method considered here, in the context of the solution of the Euler equations has been performed by Tuminaro[139]; however, to date, no concrete parallel implementation and further evaluation of the method has been done in the context of compressible flow calculations. Here we propose a detailed analysis of the method and its evaluation in the context of the parallel solution of the two-dimensional Euler equations on unstructured meshes.

The rest of the section is organized as follows. In subsection 3.2.2, we first introduce the multiplicative and additive multigrid algorithms in a general setting; then we outline the characteristics of the starting-point monogrid Euler solver and we conclude the subsection by recalling the main characteristics of the multigrid by volume agglomeration strategy adopted in this study. Subsection 3.2.3 briefly recalls the parallelization strategy for the monogrid solver and details concrete parallel implementations of the multiplicative and additive multigrid algorithms. The objective of subsection 3.2.4 is to evaluate the proposed parallel multigrid algorithms through numerical simulations of two selected flows around a `NACA0012` airfoil geometry. These numerical simulations are performed on a cluster of PCs. An important remark concerning the results reported in this subsection is that the objective is not to provide the reader with a scalability analysis of multigrid methods on highly parallel MIMD computing platforms (see [94] for multigrid acceleration of complex flows); instead, we target small or medium size cluster computing systems based on high speed network interconnection and we try to demonstrate the main advantages of the proposed additive multigrid method in this context. Finally, subsection 3.2.5 concludes this study.

3.2.2 Discretization method and multigrid solution method

3.2.2.1 Numerical framework

Let $\Omega \subset \mathbb{R}^2$ be the computational domain of interest and Γ its boundary. We consider a triangulation $\mathcal{T}_h$ of the domain Ω and an associated finite element space $\mathcal{U}_h$ consisting of functions that are piecewise linear and continuous on a given element of $\mathcal{T}_h$. We assume that the space and time integration of the original set of partial differential equations has resulted in the following linear system that must be solved to advance the solution at each time step:

$$A_h u_h = f_h \tag{3.1}$$

In subsection 3.2.2.3 we derive such a system in the context of the numerical solution of the Euler equations. In that case, the matrix A_h is non-symmetric and non-definite; however, for the abstract convergence analysis of

subsection 3.2.2.2.4, we assume that A_h is a symmetric positive definite matrix. In the following we will make use of two discretizations of the domain Ω: a *fine* discretization associated to the characteristic dimension h (the initial finite element discretization), and a *coarse* discretization associated to the characteristic dimension H.

The approximation spaces associated to the fine grid $\mathcal{T}_h$ are denoted by $\mathcal{U}_h$ (space of unknowns) and $\mathcal{F}_h$ (space of right hand sides). Similarly, we denote by $\mathcal{U}_H$ and $\mathcal{F}_H$ the approximation spaces associated to the coarse grid triangulation $\mathcal{T}_H$. Multigrid algorithms rely on inter-grid transfer operators: the restriction operator is denoted by $I_{\mathcal{F}_h}^{\mathcal{F}_H}$ with $I_{\mathcal{F}_h}^{\mathcal{F}_H} : \mathcal{F}_h \longrightarrow \mathcal{F}_H$ and the prolongation operator by $I_{\mathcal{U}_H}^{\mathcal{U}_h}$ with $I_{\mathcal{U}_H}^{\mathcal{U}_h} : \mathcal{U}_H \longrightarrow \mathcal{U}_h$. Let $A_H : \mathcal{U}_H \longrightarrow \mathcal{F}_H$ be the coarse grid operator and $f_H \in \mathcal{F}_H$ such that the equation:

$$A_H u_H = f_H \tag{3.2}$$

stands for the discrete formulation on $\mathcal{T}_H$. The variational approach for the construction of the coarse grid operator in (3.2) is given by:

$$A_H = I_{\mathcal{F}_h}^{\mathcal{F}_H} A_h I_{\mathcal{U}_H}^{\mathcal{U}_h} \quad \text{with} \quad I_{\mathcal{F}_h}^{\mathcal{F}_H} = c(I_{\mathcal{U}_H}^{\mathcal{U}_h})^* \tag{3.3}$$

Details about this definition of A_H, $I_{\mathcal{F}_h}^{\mathcal{F}_H}$ and $I_{\mathcal{U}_H}^{\mathcal{U}_h}$ can be found in [143]. Another alternative which is adopted in this study is to define A_H from the discretization of the continuous problem on the coarse grid levels. In the following, we introduce *multiplicative* and *additive* ideal two-grid algorithms for the solution of the linear system (3.1).

3.2.2.2 Multiplicative and additive linear multigrid algorithms

3.2.2.2.1 Ideal two-grid algorithm, multiplicative formulation. The classical ideal two-grid algorithm consists of the following steps:

- *fine grid pre-smoothing:*

$$u_h^{(n+\frac{1}{2})} = L^{\nu_1} u_h^{(n)} + g_1 \tag{3.4}$$

  where L is the smoothing operator, g_1 is a vector depending on the right-hand side f_h, and ν_1 is the number of pre-smoothing steps.

- *coarse grid correction.* The error equation on the coarse grid is stated using the fine grid residual after the pre-smoothing step:

$$A_H c_H = r_H^{(n+\frac{1}{2})} \quad \text{with} \quad r_H^{(n+\frac{1}{2})} = I_{\mathcal{F}_h}^{\mathcal{F}_H} \left(f_h - A_h (L^{\nu_1} u_h^{(n)} + g_1) \right) \tag{3.5}$$

  In the ideal two-grid algorithm the above residual is restricted on the coarse grid and the resulting system is solved exactly. The fine grid correction is obtained by prolongating the solution of the coarse grid equation:

$$c_h = I_{\mathcal{U}_H}^{\mathcal{U}_h} A_H^{-1} I_{\mathcal{F}_h}^{\mathcal{F}_H} r_h^{(n+\frac{1}{2})} \tag{3.6}$$

- *fine grid post-smoothing and solution update:*

$$u_h^{(n+1)} = L^{\nu_2} (u_h^{(n+\frac{1}{2})} + c_h) + g_2 \tag{3.7}$$

  where ν_2 is the number of post-smoothing steps.

The iterative error is given by $e^{(n)} = u_h^{(n)} - u^\star$ where $u_h^{(n)}$ denotes the approximate solution of equation (3.1) on the fine grid after the n^{th} two-grid cycle; $u^\star$ is the exact solution. Using the consistency relation $u^\star = L^{\nu_i} u^\star + g_i$ [18], we deduce the expression for the evolution of the iterative error during one cycle of the ideal two-grid algorithm:

$$e_h^{(n+1)} = M^M e_h^{(n)} \quad \text{with} \quad M^M = L^{\nu_2}(\mathsf{Id} - I_{\mathcal{U}_H}^{\mathcal{U}_h} A_H^{-1} I_{\mathcal{F}_h}^{\mathcal{F}_H} A_h) L^{\nu_1} \tag{3.8}$$

Eq. (3.8) clearly shows the *multiplicative* nature of the classical formulation of the ideal two-grid algorithm : the coarse grid correction step depends on the result of the pre-smoothing steps, and the post-smoothing steps act on the corrected system.

3.2.2.2.2 Ideal two-grid algorithm, additive formulation. The basic idea of the *additive* formulation initially proposed by Chan and Tuminaro[30] is to use filtering techniques in order to isolate high frequency (HF) from low frequency (LF) modes of the residual. This process can be mathematically stated by introducing orthogonal projection operators that are applied to the fine grid residual. We note in passing that the resulting algorithm belongs to the class of additive subspace correction (ASC) methods introduced by Xu[148]. In order to be more general, the notion of residual filtering is here extended to correction filtering. This setting is particularly useful for the abstract convergence analysis detailed in subsection 3.2.2.2.4. The problem at hand is again given by eq. (3.1). The new version of the ideal two-grid algorithm consists of the following steps:

- *evaluation of the fine grid residual:*

$$r_h^{(n)} = f_h - A_h u_h^{(n)} \tag{3.9}$$

  The residual filtering technique is applied to the fine grid residual. In the additive formulation some of the characteristics of the classical multigrid algorithm, which are closely related to its multiplicative nature, are lost. For instance, there is no definition of cycle (V-cycle, F-cycle or W-cycle) and the notions of pre- and post- smoothing steps are simply replaced by smoothing operations on the high frequency modes subsystem.

- *residual filtering:*

$$r_h^1 = T_h r_h^{(n)} \quad \text{and} \quad r_h^2 = (\mathsf{Id} - T_h) r_h^{(n)} = S_h r_h^{(n)} \tag{3.10}$$

  The role of the filtering operators T_h and S_h is to operate on a decomposition of the initial residual in terms of high and low frequency modes. The properties of these operators must ensure that $r_h^{(n)} = r_h^1 + r_h^2$. This allows the definition of two disconnected error equations $A_h c_h^1 = r_h^1$ and $A_h c_h^2 = r_h^2$ that can be solved concurrently:

  - *high frequency problem, smoothing step:*

$$c_h^1 = (\mathsf{Id} - L^{\nu_{add}}) A_h^{-1} r_h^1 \tag{3.11}$$

    Eq. (3.11) states the application of ν_{add} iterations of the smoother L that is $c_h^{1,(\nu_{add})} = L^{\nu_{add}} c_h^{1,(0)} + g$. Then, eq. (3.11) results from $c_h^{1,(0)} = 0$ and the fact that the smoother verifies the consistency relation $g = (\mathsf{Id} - L^{\nu_{add}}) A_h^{-1} r_h^1$. Note that the smoother is not applied to the solution vector, as it is the case in the classical multigrid formulation, but to a residual;

  - *low frequency problem, coarse grid correction:*

$$c_h^2 \quad = I_{\mathcal{U}_H}^{\mathcal{U}_h} A_H^{-1} I_{\mathcal{F}_h}^{\mathcal{F}_H} r_h^2 \tag{3.12}$$

- *correction filtering:*

$$c_h^I = \overline{T}_h c_h^1 \quad \text{and} \quad c_h^{II} = (\mathsf{Id} - \overline{T}_h) c_h^2 = \overline{S}_h c_h^2 \tag{3.13}$$

The theoretical objective of applying filtering operators to the corrections is to ensure that the latter are not polluted by spurious frequencies and thus to enforce that these corrections belong to distinct subspaces.

- *solution update:* $u_h^{(n+1)} = u_h^{(n)} + c_h^I + c_h^{II}$

The iterative error is now subjected to the relations:

$$\begin{array}{c} e_h^{(n+1)} = M^A e_h^{(n)} \quad \text{with} \quad M^A = \mathsf{Id} - (D_h + E_h) A_h \\ \text{where} \quad D_h = \overline{T}_h (\mathsf{Id} - L^{\nu_{add}}) A_h^{-1} T_h \quad \text{and} \quad E_h = \overline{S}_h I_{\mathcal{U}_H}^{\mathcal{U}_h} A_H^{-1} I_{\mathcal{F}_h}^{\mathcal{F}_H} S_h \end{array} \tag{3.14}$$

It is clear from eq. (3.14) that the operations performed on the fine grid level (related to D_h) and on the coarse grid level (related to E_h) are independent.

3.2.2.2.3 Filtering operators. We consider that the vector space $\mathcal{F}_h$ is spanned by the usual orthogonal Fourier basis in which the high and low frequency (HF and LF) modes of a given vector quantity are distinguished. This identification permits us to decompose this vector space as the direct sum $\mathcal{F}_h = \mathcal{G}_h \oplus \mathcal{G}_h^\perp$ in which the subspace $\mathcal{G}_h$ is assumed to support the HF modes and $\mathcal{G}_h^\perp$ is the orthogonal supplementary subspace. Then, the filtering of the HF modes of the residual can be stated as an orthogonal projection (in terms of the Euclidean dot product) of the residual $r_h^{(n)} \in \mathcal{F}_h$ onto $\mathcal{G}_h$, that is $T_h : \mathcal{F}_h \longrightarrow \mathcal{G}_h$. For consistency reasons, it is mandatory that each mode be treated at most once, either as a HF mode or as a LF one. In other words, the subspace associated to the LF modes of the residual $\mathcal{G}_h^\perp \subset \mathcal{F}_h$ is induced by the operator $S_h : \mathcal{F}_h \longrightarrow \mathcal{G}_h^\perp$ which is an orthogonal projection from $\mathcal{F}_h$ onto $\mathcal{G}_h^\perp$ and such that $S_h = I - T_h$.

For a general formulation of the additive algorithm we need another couple of filtering operators to decompose the HF modes from the LF modes of the correction. However, we also add the constraint that the HF modes of the residual are also HF modes of the correction. This means that the filtering operators for the residual and the correction must have the same properties and result in the same effects. In theory, the only difference in the definition of these operators comes with the spaces and subspaces involved. As previously , we consider a decomposition of the vector space $\mathcal{U}_h$ as $\mathcal{U}_h = \mathcal{V}_h \oplus \mathcal{V}_h^\perp$. We recall that we aim at applying filtering operators on the problem $A_h c_h = r_h$, where c_h and r_h respectively denote the correction and the residual. Therefore, if T_h is the filtering operator applied to the residual, then the filtering operator for the correction $\overline{T}_h$ must satisfy: $A_h \overline{T}_h c_h = T_h r_h$. This yields a definition of $\overline{T}_h : \mathcal{U}_h \longrightarrow \mathcal{V}_h$ as:

$$\overline{T}_h = A_h^{-1} T_h A_h \tag{3.15}$$

Since T_h is an orthogonal projection, we wish that $\overline{T}_h$ be also an orthogonal projection, this time from $\mathcal{U}_h$ onto $\mathcal{V}_h$. However, if T_h is a projection in terms of the Euclidean dot product, the dot product defining $\overline{T}_h$ has to be determined. The reader can verify that the dot product $< ., . >_{A_h^2}$ allows to express $\overline{T}_h$ as an orthogonal projection from $\mathcal{U}_h$ onto $\mathcal{V}_h$. Indeed, for a symmetric positive definite operator B, we have that P is an orthogonal projection if and only if:

$$\begin{array}{c} < Px \ , \ (\mathsf{Id} - P)x >_B = 0 \quad \forall x \\ \Leftrightarrow \quad (Px)^\star B (\mathsf{Id} - P) x = 0 \quad \Leftrightarrow \quad P^\star B (\mathsf{Id} - P) = 0 \end{array} \tag{3.16}$$

If $B = \mathsf{Id}$ (that is the Euclidean dot product is selected) then we obtain that $P = P^\star P$ that is $P^\star = P$. It is then clear that the property given in eq. (3.16) is verified for $P = \overline{T}_h$ as defined in eq. (3.15) and with $B = A_h^2$. Following what has been done for the filtering of the residual, we define the filtering operator for the LF modes of the correction by $\overline{S}_h = A_h^{-1} S_h A_h = \mathsf{Id} - \overline{T}_h$ where $\overline{S}_h$ is an orthogonal projection from $\mathcal{U}_h$ onto $\mathcal{V}_h^\perp$.

3.2.2.2.4 Abstract convergence analysis. We propose here an abstract convergence analysis which is inspired from the theory of Hackbusch[68] for classical multigrid algorithms. In particular, Hackbusch[68] demonstrates that the convergence of the multiplicative ideal two-grid algorithm is subjected to smoothing and approximation properties. These properties are a characterization of the role of the pre- and post-smoothing steps on the fine grid, and of the role of the coarse grid correction. In the case of the additive ideal two-grid algorithm, the smoothing and approximation properties are expressed in forms slightly different from the usual formulations[68]. Let W_i be a Hilbert space with $\langle .,. \rangle_{W_i}$ and $\|.\|_{W_i}$ denoting the associated dot product and norm. The norm of an operator $H : W_i \longrightarrow W_j$ is defined by:

$$\|H\|_{W_j \leftarrow W_i} = \sup_{x \in W_i,\ \|x\|_{W_i} \neq 0} \left(\frac{\|Hx\|_{W_j}}{\|x\|_{W_i}} \right) \tag{3.17}$$

In the present context, the smoothing property has the following form:

$$\begin{cases} \|A_h L^\nu\|_{\mathcal{F}_h \leftarrow \mathcal{U}_h} \leq \eta(\nu) h^{-1} \ , \quad \forall \nu,\ 1 \leq \nu < \overline{\nu}(h) \ , \\ \eta(\nu) \rightarrow 0 \text{ when } \ \nu \rightarrow \infty \ \text{ and } \ \overline{\nu}(h) \rightarrow \infty \ \text{ when } \ h \rightarrow 0 \end{cases} \tag{3.18}$$

As mentioned in [68], eq. (3.18) can be factorized using A_h^{-1}. Since the iterative operator L is such that $L^\nu : \mathcal{U}_h \longrightarrow \mathcal{U}_h$, we deduce the following form of the smoothing property:

$$\left\{ \begin{array}{l} \|L^\nu\|_{\mathcal{U}_h \leftarrow \mathcal{U}_h} \leq \eta(\nu) \ , \quad \forall \nu, \ 1 \leq \nu < \overline{\nu}(h) \\ \eta(\nu) \rightarrow 0 \ \text{ when } \ \nu \rightarrow \infty \ \text{ and } \ \overline{\nu}(h) \rightarrow \infty \ \text{ when } \ h \rightarrow 0 \end{array} \right. \tag{3.19}$$

As previously, we wish to use a non-classical form of the approximation property. The same type of factorization as applied to the smoothing property applied to the classical form of the approximation property [68] gives:

$$\|\mathsf{Id} - I_{\mathcal{U}_H}^{\mathcal{U}_h} A_H^{-1} I_{\mathcal{F}_h}^{\mathcal{F}_H} A_h\|_{\mathcal{U}_h \leftarrow \mathcal{U}_h} \leq C \tag{3.20}$$

Here C is a constant. A necessary condition to enforce this property is to define A_H, $I_{\mathcal{F}_h}^{\mathcal{F}_H}$ and $I_{\mathcal{U}_H}^{\mathcal{U}_h}$ such that eq. (3.3) is verified. From now, we assume that properties (3.19) and (3.20) are verified by the various operators taking part in the definition of the additive ideal two-grid algorithm. Clearly, the following relation holds for the iterative error:

$$\|e_h^{(n+1)}\|_{\mathcal{U}_h} = \|M^A e_h^{(n)}\|_{\mathcal{U}_h} \tag{3.21}$$

The iterative error is decomposed into HF and LF modes using the $\overline{T}_h$ and $\overline{S}_h$ filtering operators:

$$\|e_h^{(n+1)}\|_{\mathcal{U}_h} \leq \|\overline{T}_h M^A e_h^{(n)}\|_{\mathcal{U}_h} + \|\overline{S}_h M^A e_h^{(n)}\|_{\mathcal{U}_h} \tag{3.22}$$

Using (3.14) we have:

$$\begin{aligned} \overline{T}_h M^A &= \overline{T}_h - \overline{T}_h (D_h + E_h) A_h = \overline{T}_h - \overline{T}_h D_h A_h \\ &= \overline{T}_h - \overline{T}_h^2 (\mathsf{Id} - L^{\nu_{add}}) A_h^{-1} T_h A_h \\ &= \overline{T}_h L^{\nu_{add}} \overline{T}_h \end{aligned} \tag{3.23}$$

where we have used the relations $\overline{T}_h^2 = \overline{T}_h$ and $\overline{T}_h \overline{S}_h = 0$ and the definition (3.15) of $\overline{T}_h$. Similarly, we have:

$$\begin{aligned} \overline{S}_h M^A &= \overline{S}_h - \overline{S}_h (D_h + E_h) A_h = \overline{S}_h - \overline{S}_h E_h A_h \\ &= \overline{S}_h - \overline{S}_h^2 I_{\mathcal{U}_H}^{\mathcal{U}_h} A_H^{-1} I_{\mathcal{F}_h}^{\mathcal{F}_H} S_h A_h \\ &= \overline{S}_h (\mathsf{Id} - I_{\mathcal{U}_H}^{\mathcal{U}_h} A_H^{-1} I_{\mathcal{F}_h}^{\mathcal{F}_H} A_h) \overline{S}_h \end{aligned} \tag{3.24}$$

We deduce from the above expressions that:

$$\|e_h^{(n+1)}\|_{\mathcal{U}_h} = \|\overline{T} L^{\nu_{add}} \overline{T_h} e_h^{(n)}\|_{\mathcal{U}_h} + \|\overline{S}_h (\mathsf{Id} - I_{\mathcal{U}_H}^{\mathcal{U}_h} A_H^{-1} I_{\mathcal{F}_h}^{\mathcal{F}_H} A_h) \overline{S}_h e_h^{(n)}\|_{\mathcal{U}_h} \tag{3.25}$$

and:

$$\begin{aligned} \|e_h^{(n+1)}\|_{\mathcal{U}_h} &\leq \|\overline{T_h}\|_{\mathcal{U}_h} \ \|L^{\nu_{add}}\|_{\mathcal{U}_h} \ \|\overline{T}_h e_h^{(n)}\|_{\mathcal{U}_h} \\ &+ \|\overline{S}_h\|_{\mathcal{U}_h} \ \|\mathsf{Id} - I_{\mathcal{U}_H}^{\mathcal{U}_h} A_H^{-1} I_{\mathcal{F}_h}^{\mathcal{F}_H} A_h\|_{\mathcal{U}_h} \ \|\overline{S}_h e_h^{(n)}\|_{\mathcal{U}_h} \end{aligned}$$

The smoothing and approximation properties together with the fact that $\|\overline{S}_h\|_{\mathcal{U}_h} = \|\overline{T}_h\|_{\mathcal{U}_h} = 1$ give:

$$\|e_h^{(n+1)}\|_{\mathcal{U}_h} \leq \eta(\nu_{add}) \|\overline{T}_h e_h^{(n)}\|_{\mathcal{U}_h} + C \|\overline{S}_h e_h^{(n)}\|_{\mathcal{U}_h}$$

and:

$$\|e_h^{(n+1)}\|_{\mathcal{U}_h} \leq \max\Big(\eta(\nu_{add}), C\Big) \ \Big(\|\overline{T}_h e_h^{(n)}\|_{\mathcal{U}_h} + \|\overline{S}_h e_h^{(n)}\|_{\mathcal{U}_h} \Big)$$

The filtering operators decompose the space of corrections into the direct sum of two subspaces so that we can finally deduce the following estimation:

$$\|e_h^{(n+1)}\|_{\mathcal{U}_h} \leq \max\left(\eta(\nu_{add}), C\right) \|e_h^{(n)}\|_{\mathcal{U}_h} \tag{3.26}$$

If $\max(\eta(\nu_{add}), C) < 1$, the additive ideal two-grid algorithm is convergent. We note that this condition is more restrictive than the condition obtained for the multiplicative form of the ideal two-grid algorithm which requires that $C\eta(\nu_{mul}) < 1$ (see [68] for more details). Nevertheless, the additive formulation of the ideal two-grid algorithm preserves the mesh independent convergence property already enjoyed by the multiplicative algorithm.

3.2.2.3 Monogrid solution of the Euler calculations

The conservative form of the Euler equations in the two-dimensional case is given by:

$$\frac{\partial W}{\partial t} + \vec{\nabla}.\vec{\mathbb{F}}^c(W) = 0 \quad , \quad W = \left(\rho \ , \ \rho\vec{V} \ , \ E\right)^T \quad , \quad \vec{\nabla} = \left(\frac{\partial}{\partial x} \ , \ \frac{\partial}{\partial y}\right)^T \tag{3.27}$$

In eq. (3.27), $W = W(\vec{x}, t)$ where $\vec{x}$ and t respectively denote the spatial and temporal variables. The components of the conservative flux $\vec{\mathbb{F}}^c(W) = (F_x(W), F_y(W))^T$ write as:

$$F_x(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix} \quad , \quad F_y(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}$$

In the above expressions, ρ is the density, $\vec{V} = (u \ , \ v)^T$ is the velocity vector, E is the total energy per unit of volume and p is the pressure. The pressure is deduced from the other variables using the state equation for a perfect gas:

$$p = (\gamma_e - 1)(E - \frac{1}{2}\rho \parallel \vec{V} \parallel^2)$$

where γ_e is the ratio of specific heats ($\gamma_e = 1.4$ for the air).

The Euler equations (3.27) are discretized in space by using the mixed element/volume formulation on unstructured triangular meshes which is described in section 1.2.1 of chapter 1. Time integration of the resulting semi-discrete equations makes use of the linearized implicit scheme described in section 1.4.1. Then, at each pseudo-time step, a linear system must be solved to advance the solution in time.

3.2.2.4 Multigrid by volume agglomeration

The multigrid strategy adopted in this study is based on the use of macro elements which form the coarse discretizations of the computational domain. This so-called multigrid by volume agglomeration method is described in details in subsection 3.1.3 of section 3.1 in the context of 3D flows. Here, we simply discussed those aspects that are relevant to the present study.

3.2.2.4.1 Inter-grid transfer operators. From the description of the additive multigrid algorithm in subsection 3.2.2.2.2, we need to define two prologation operators: $I_{\mathcal{U}_H}^{\mathcal{U}_h} : \mathcal{U}_H \rightarrow \mathcal{U}_h$ is used to transfer solution or correction vectors, while $I_{\mathcal{F}_H}^{\mathcal{F}_h} : \mathcal{F}_H \rightarrow \mathcal{F}_h$ acts on right-hand sides or residuals. The prolongation operator $I_{\mathcal{F}_H}^{\mathcal{F}_h}$ is normally not used in the classical (multiplicative) formulation of the multigrid algorithm. However, as it will be discussed in subsection 3.2.2.5, the construction of the filtering operators that are inherent to the additive multigrid formulation makes use of this prolongation operator. In practice, we build the prolongation operator from a canonical injection (see figure 3.19) which is used for the transfer of both types of quantity. Let $x(j)_h \in \mathcal{U}_h$ (respectively $y(j)_h \in \mathcal{F}_h$) and $x(J)_H \in \mathcal{U}_H$ (respectively $y(J)_H \in \mathcal{F}_H$) where j stands for the cell C_j of the fine grid, which is included in cell C_J (denoted by J) of the coarse grid.

The prolongation operator for a solution/correction vector is defined by:

$$x(j)_h = (1-\omega)\ x(J)_H \ + \ \omega \ \frac{\sum\limits_{k\in\mathcal{N}(j)} \mathcal{A}(k)_h \ x(K)_H}{\sum\limits_{k\in\mathcal{N}(j)} \mathcal{A}(k)_h} \tag{3.28}$$

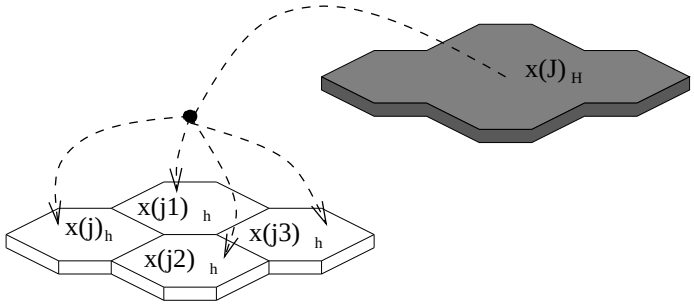


Figure 3.19: The prolongation operator on agglomerated grids

In eq. (3.28), $\mathcal{A}(j)_h$ denotes the area of cell C_j and $\mathcal{N}(j)$ is the set of cells C_k that are direct neighbors of C_j. The prolongation operator for transferring the residual is defined by:

$$y(j)_h = \mathcal{A}(j)_h \left((1-\omega)\ \frac{y(J)_H}{\mathcal{A}(J)_H} \ + \ \omega \ \frac{\sum\limits_{k\in\mathcal{N}(j)} \mathcal{A}(k)_h \ \frac{y(K)_H}{\mathcal{A}(K)_H}}{\sum\limits_{k\in\mathcal{N}(j)} \mathcal{A}(k)_h} \right) \tag{3.29}$$

In eq. (3.28) and eq. (3.29), ω is a weighting parameter which is used to implement a combination between the canonical injection ($\omega = 0$) and an averaging operation ($\omega = 1$) on neighboring cells of the injected values where K denotes the coarse grid cell C_K to which the fine grid cell C_k belongs.

Figure 3.20 illustrates the role of a restriction operator in the context of agglomerated grids. As previously, we define two restriction operators: $I_{\mathcal{U}_h}^{\mathcal{U}_H} : \mathcal{U}_h \rightarrow \mathcal{U}_H$ and $I_{\mathcal{F}_h}^{\mathcal{F}_H} : \mathcal{F}_h \rightarrow \mathcal{F}_H$.

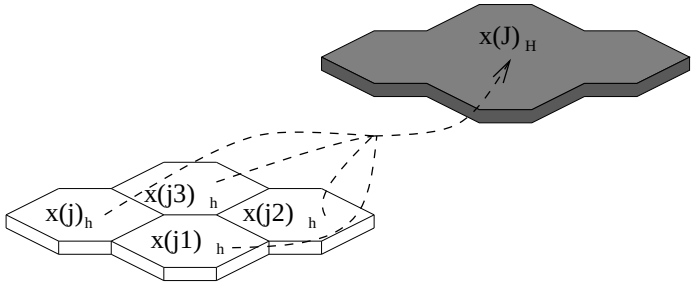


Figure 3.20: The restriction operator on agglomerated grids

If we decompose the prolongation operator using a simple canonical injection ($I_{\mathcal{U}_h}^{\mathcal{U}_H}$) and an averaging operator (L) then we can define the restriction operator as the adjoint of the prolongation operator:

$$\langle I_{\mathcal{U}_H}^{\mathcal{U}_h} L^* x_h, x_H \rangle_{\mathcal{U}_H} = \langle x_h, L I_{\mathcal{U}_h}^{\mathcal{U}_H} x_H \rangle_{\mathcal{U}_h} \tag{3.30}$$

where:

$$\left\{\begin{array}{lcl} \langle x_h, y_h\rangle_{\mathcal{U}_h} & = & \displaystyle\sum_{j=1}^{dim(\mathcal{U}_h)} x(j)_h\ y(j)_h\ \mathcal{A}(j)_h \\ \langle x_H, y_H\rangle_{\mathcal{U}_H} & = & \displaystyle\sum_{J=1}^{dim(\mathcal{U}_H)} x(J)_H\ y(J)_H\ \mathcal{A}(J)_H \end{array}\right. \tag{3.31}$$

In eq. (3.30), L^* is an averaging operator corresponding to the adjoint of L. From eq. (3.30) and eq. (3.28) we deduce the expression of the restriction operator:

$$x(J)_H = \frac{1}{\mathcal{A}(J)_H} \sum_{j\in\mathcal{C}(J)} \mathcal{A}(j)_h x(j)_h \tag{3.32}$$

where $\mathcal{C}(J)$ denotes the set of fine grid cells constituting the coarse grid cell C_J. Eq. (3.32) gives the expression of the adjoint of the canonical injection. For the averaging operator L^* we get:

$$L^*x(j)_h \ = \ (1-\omega)\ x(j)_h \ + \ \omega \sum_{k\in\mathcal{N}(j)} \frac{\mathcal{A}(k)_h x(k)_h}{\displaystyle\sum_{l\in\mathcal{N}(k)} \mathcal{A}(l)_h} \tag{3.33}$$

where $\mathcal{N}(k)$ denotes the set of cells C_l that are direct neighbors of cell C_k. We remark that the prolongation operators defined by eq. (3.28) and eq. (3.29) differ from a multiplicative coefficient. It is easy to express $I_{\mathcal{F}_H}^{\mathcal{F}_h}$ in terms of $I_{\mathcal{V}_H}^{\mathcal{U}_h}$. Let $\mathcal{A}_h$ and $\mathcal{A}_H$ denote diagonal matrices such that $(\mathcal{A}_h)_{j,j} = \mathcal{A}(j)_h$ and $(\mathcal{A}_H)_{J,J} = \mathcal{A}(J)_H$; then we have:

$$I_{\mathcal{F}_H}^{\mathcal{F}_h} \ = \ \mathcal{A}_h\ I_{\mathcal{U}_H}^{\mathcal{U}_h}\ \mathcal{A}_H^{-1} \ \text{ and } \ I_{\mathcal{F}_h}^{\mathcal{F}_H} \ = \ \mathcal{A}_H\ I_{\mathcal{U}_h}^{\mathcal{U}_H}\ \mathcal{A}_h^{-1} \tag{3.34}$$

3.2.2.5 Filtering operators

In the additive formulation, the filtering technique aims at operating a separation between HF and LF modes. In some sense, the multiplicative multigrid method is implicitly using some type of filtering: the pre-smoothing step on the fine grid system is responsible for damping the HF modes of the error; as these latter frequencies are generally not completely removed by the pre-smoothing operation, the restriction operator is used to ensure that only the fine grid LF modes are treated on the coarse grid. This very simple interpretation of the behavior of the multiplicative multigrid method clearly shows that the inter-grid transfer operators can be used as building blocks for the filtering operators. Indeed, the design of the filtering operators using the inter-grid transfer operators has been advocated by Tuminaro[139] and can be stated as:

$$\left\{\begin{array}{lcllcl} S_h & = & I_{\mathcal{F}_H}^{\mathcal{F}_h}\ I_{\mathcal{F}_h}^{\mathcal{F}_H} & \text{and} & T_h & = & I \ - \ S_h \ = \ I \ - \ I_{\mathcal{F}_H}^{\mathcal{F}_h}\ I_{\mathcal{F}_h}^{\mathcal{F}_H} \\ \overline{S}_h & = & I_{\mathcal{U}_H}^{\mathcal{U}_h}\ I_{\mathcal{U}_h}^{\mathcal{U}_H} & \text{and} & \overline{T}_h & = & I \ - \ \overline{S}_h \ = \ I \ - \ I_{\mathcal{U}_H}^{\mathcal{U}_h}\ I_{\mathcal{U}_h}^{\mathcal{U}_H} \end{array}\right. \tag{3.35}$$

At this point, an important remark is that the filtering operators defined by (3.35) and the inter-grid transfer operators of subsection 3.2.2.4.1, are computationally cheap however they do not strictly define projection operators (see section 3.2.4). In practice we use $\omega = 1$ in (3.28) and (3.29) and $\omega = 0$ in (3.33) .

3.2.3 Parallel computing aspects

The parallelization strategy adopted for the standard (multiplicative) multigrid solver is very similar to what has bee done in the three-dimensional case (see subsection 3.1.4 of section 3.1). More details are given below for what concern the parallel implementation of the additive multigrid method.

The abstract convergence analysis of subsection 3.2.2.2.4 has shown that the additive multigrid algorithm is convergent but less efficient than the classical multiplicative formulation. This is the first point to take into

consideration for a concrete implementation (sequential or parallel) of the additive multigrid formulation. The second point is specifically concerned with the parallel implementation. Indeed, preliminary experiments using a domain partitioning based parallel implementation of the multiplicative multigrid algorithm have shown that on distributed memory platforms with relatively high communication overheads, such as a cluster of PCs, the communication costs generally represent between 10% to 20% of the total execution times, depending on the number of processors and on the number of coarse grid levels. Meanwhile, the same code running on a shared memory MIMD system demonstrates communication costs of the order of 5% in the worst cases[25]. From these results, it is clear that one cannot reasonably expect a notable gain in parallel efficiency with the additive multigrid algorithm on shared memory platforms. In this study, we target computing platforms of the first type. However, the distributed memory paradigm that characterizes the parallelization strategy described previously is not to facilitate the parallel implementation of the additive formulation. Applying directly an *inter-level* parallelization by allowing each grid level to be treated by a different group of processes (the *intra-level* parallelization taking place within each group) would require huge amounts of communication to distribute the data at initialization, as well as to recombine results during the linear system solution step. Such a strategy would be more adapted to a shared memory parallel implementation. Therefore, the approach adopted here relies on and extends the strategy used for the parallelization of the multiplicative algorithm.

For the preliminary implementation considered here, we have decided to limit the application of the additive formulation to the two coarsest grid levels of the multigrid hierarchy. Clearly, this results in a hybrid multiplicative/additive multigrid algorithm. In practice, the smoothing steps on the two coarsest grid levels are still performed on a submesh basis as for the other (multiplicative) grid levels and thus, incur point to point communications steps at submesh interfaces. However, the corresponding interface values are now packed into a single message which is exchanged prior to performing the smoothing steps for the two correction systems associated to the additive grid levels. Figure 3.21 illustrates the parallel implementation of this hybrid multiplicative/additive multigrid formulation. The gain in parallel efficiency with such an approach will come from the fact that most of the time spent in communication is due to the initialization step (that is the communication latency), especially for short message sizes as it is the case for the coarsest grid levels.

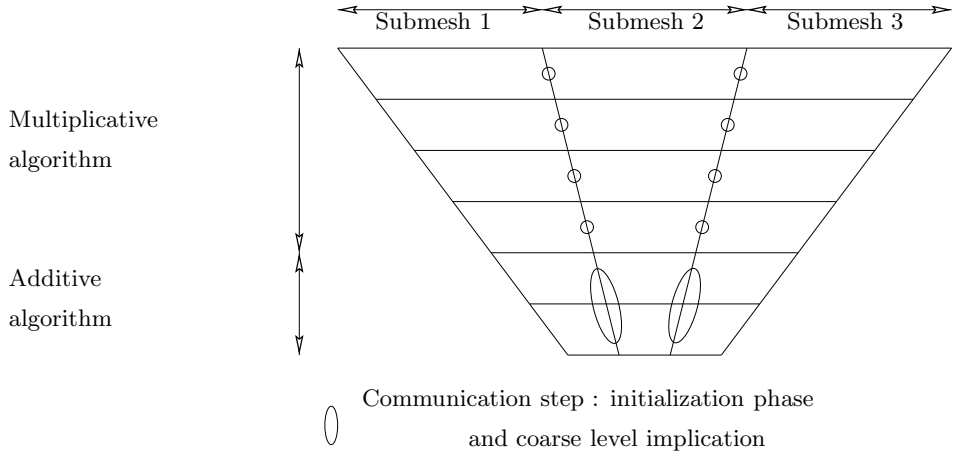


Figure 3.21: Implementation of the hybrid multiplicative/additive multigrid algorithm

3.2.4 Numerical and performance results

3.2.4.1 Test cases definition

The selected test cases are concerned with the calculation of steady external flows around a NACA0012 airfoil. Three unstructured triangular meshes have been used (see table 3.8).

Table 3.8: Characteristics of the NACA0012 airfoil meshes

Mesh	# Vertices	# Triangles	# Edges
N1	3114	6056	9170
N2	12284	24224	36508
N3	48792	96896	145688

The following situations have been considered:

S1: the transonic flow at a free stream Mach number equal to 0.85 and an angle of attack of 0°. In that case, the time step is obtained using the rule CFL=$20 \times i_t$ where i_t denotes the pseudo-time iteration. The Van Albada limiter is used in the MUSCL technique (see Fezoui and Dervieux[52]).

S2: the subsonic flow at a free stream Mach number equal to 0.3 and an angle of attack of 0°. In that case, the extension to second order accuracy in space does not use any limiting procedure. The time step is again obtained using the law CFL=$20 \times i_t$.

3.2.4.2 Computing platforms and conventions

Numerical experiments have been performed on a cluster of 16 Pentium Pro/200 Mhz PCs running the Linux system and interconnected via a 100 Mbit/s FastEthernet switch. The MPI implementation is MPICH. The code is written in Fortran 77 and the GNU G77 compiler has been used with maximal optimization options. Performance results are given for 64 bit arithmetic computations. In the following tables, N_p is the number of processes for the parallel execution, N_g is the total number of levels in the multigrid hierarchy (fine mesh included), N_c denotes the number of multigrid cycles used for each linear system solution; "Total time" denotes the total elapsed execution time and "CPU time" denotes the total CPU time (taken as the maximum value over the local measures); "% CPU" denotes the ratio of "CPU time" to "Total time". This ratio will be our principal metric of parallel efficiency. The difference between "Total time" and "CPU time" basically consists of the sum of the communication and idle times, the latter being related to computational load unbalance. In practice, the computational load unbalance has been minimized as far as it was possible when partitioning the initial triangular meshes; however, the generation of coarse discretisations by volume agglomeration is not guaranteed to yield an optimal repartition of the computational load. Finally, the parallel speedup $S(N_p)$ is always calculated using the elapsed execution times.

3.2.4.3 Ideal two-grid algorithms

This first series of experiments aims at assessing the numerical efficiency of the multiplicative and additive ideal two-grid algorithms. In order to do so, each calculation is limited to the solution of the first linear system (i.e. the linear system resulting from the first implicit time step) using one coarse grid; the coarse grid system is fully converged using the Jacobi relaxation method which is also taken to be the smoother for the fine grid system.

3.2.4.3.1 Transonic flow test case. For each mesh of table 3.8, the following solution methods are compared:

- the multiplicative algorithm using $\nu_1 = 2$ pre-relaxations and $\nu_2 = 2$ post-relaxations (see figure 3.22 left);
- the additive algorithm based on residual filtering using $\nu_{add} = 2$ (see figure 3.22 right), $\nu_{add} = 3$ (see figure 3.23 left) and $\nu_{add} = 4$ (see figure 3.23 right) smoothing steps for the high frequency correction system;
- the additive algorithm based on residual and correction filtering using $\nu_{add} = 2$ (see figure 3.24), $\nu_{add} = 3$ (see figure 3.25 left) and $\nu_{add} = 4$ (see figure 3.25 right) smoothing steps for the high frequency correction system.

The effective number of cycles to convergence are summarized in table 3.9.

3.2.4.3.2 Subsonic flow test case. For each mesh of table 3.8, the following situations are compared:

- the multiplicative algorithm using $\nu_1 = 2$ pre-relaxations and $\nu_2 = 2$ post-relaxations (see figure 3.26 left);
- the additive algorithm based on residual filtering using $\nu_{add} = 2$ (see figure 3.26 right), $\nu_{add} = 3$ (see figure 3.27 left) and $\nu_{add} = 4$ (see figure 3.27 right) smoothing steps for the high frequency correction system;
- the additive algorithm based on residual and correction filtering using $\nu_{add} = 2$ (see figure 3.28), $\nu_{add} = 3$ (see figure 3.29 left) and $\nu_{add} = 4$ (see figure 3.29 right) smoothing steps for the high frequency correction system.

The effective number of cycles to convergence are summarized in table 3.10.

3.2.4.3.3 Comments. This first set of results calls for a number of remarks that apply to both the transonic and subsonic test cases considered here:

- first, as expected from the abstract convergence theory, the additive ideal two-grid algorithm demonstrates a convergence rate which is independent of the size of the underlying problem. This fact is particularly well verified for the case $\nu_{add} = 4$ that characterizes an additive algorithm with the same complexity (in terms of fine grid smoothing steps) as the multiplicative algorithm (i.e. $\nu_{add} = \nu_1 + \nu_2$). Indeed, for the additive algorithm based on residual filtering only, and with $\nu_{add} = 4$, the mesh size independent convergence property is always better than what is observed for the multiplicative algorithm (see the corresponding entries of tables 3.9 and 3.10);
- second, the additive algorithm using both residual and correction filtering is always less efficient than the additive algorithm which is only based on residual filtering. The main reason for this behavior is that the practical construction of the filtering operators as proposed in subsection 3.2.2.5 is not optimal in the sense that the resulting operators do not strictly define projection operators. In other words, due to this approximation, applying the correction filtering does not improve the quality of the corrections with regard to the fact that the latter should ideally belong to orthogonal subspaces;
- finally, we note that the additive algorithm with residual filtering and $\nu_{add} = 4$ is less efficient than the multiplicative algorithm with the same complexity (regarding the number of fine grid smoothing steps). Again, this observation is consistent with the result of the abstract convergence analysis.

The important point that we will retain from the previous results is that the correction filtering as constructed here does not improve the efficiency of the additive algorithm; consequently, the correction filtering is not taken into account in the next series of numerical experiments.

3.2.4.4 Multigrid algorithms

Results for the transonic and subsonic steady flows of interest are now given and compared for simulations that have been performed using mesh N3. Moreover, for all the following computations, we have used 5 grid levels (i.e. 4 coarse grid levels). The characteristics of the multiplicative and hybrid multiplicative/additive multigrid cycles that have been used for these calculations are the following:

- MUL : multiplicative V-cycle with $\nu_1 = \nu_2 = 2$ and $\nu_g = 4$.
- ADD : hybrid multiplicative/additive V-cycle with $\nu_1 = \nu_2 = 2$ for the multiplicative part and $\nu_{add} = 4$ for the additive part with residual filtering only.

We recall that for both cycles the Jacobi relaxation method is used as the smoother.

Table 3.9: Multiplicative and additive ideal two-grid algorithms
Convergence of the first linear system for the transonic flow test case

Mesh	N1	N2	N3
Multiplicative $\nu_1 = \nu_2 = 2$	12	12	16
Additive $\nu_{add} = 2$: residual filtering	24	30	32
Additive $\nu_{add} = 3$: residual filtering	18	21	23
Additive $\nu_{add} = 4$: residual filtering	17	18	19
Additive $\nu_{add} = 2$: residual and correction filtering	26	31	34
Additive $\nu_{add} = 3$: residual and correction filtering	22	23	26
Additive $\nu_{add} = 4$: residual and correction filtering	20	22	25

Table 3.10: Multiplicative and additive ideal two-grid algorithms
Convergence of the first linear system for the subsonic flow test case

Mesh	N1	N2	N3
Multiplicative $\nu_1 = \nu_2 = 2$	9	11	11
Additive $\nu_{add} = 2$: residual filtering	23	25	25
Additive $\nu_{add} = 3$: residual filtering	19	19	20
Additive $\nu_{add} = 4$: residual filtering	17	17	17
Additive $\nu_{add} = 2$: residual and correction filtering	29	29	29
Additive $\nu_{add} = 3$: residual and correction filtering	23	23	23
Additive $\nu_{add} = 4$: residual and correction filtering	21	19	21

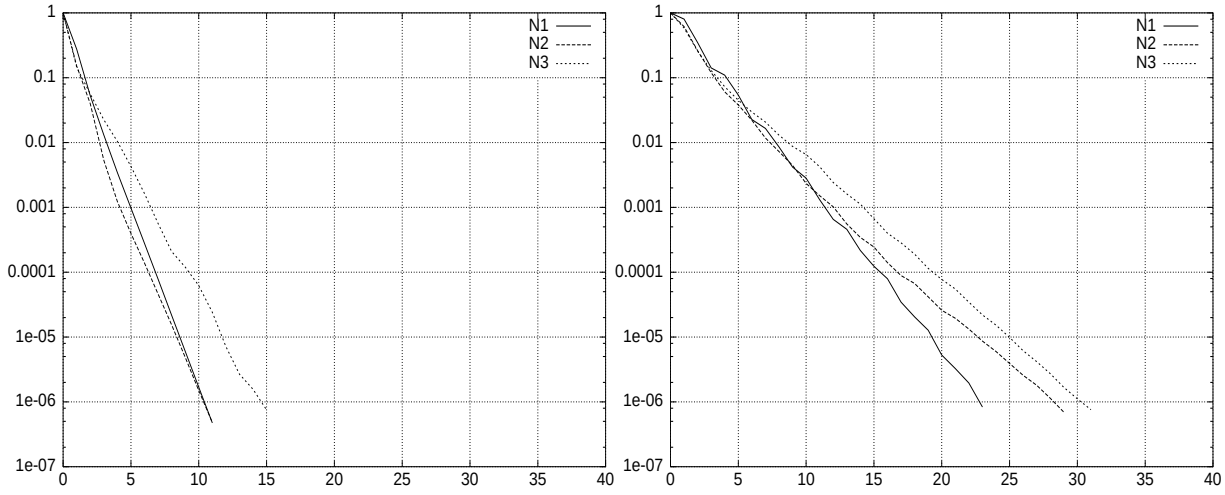


Figure 3.22: Transonic flow test case
Left : multiplicative ideal two-grid algorithm
Right : additive ideal two-grid algorithm with residual filtering, $\nu_{add} = 2$

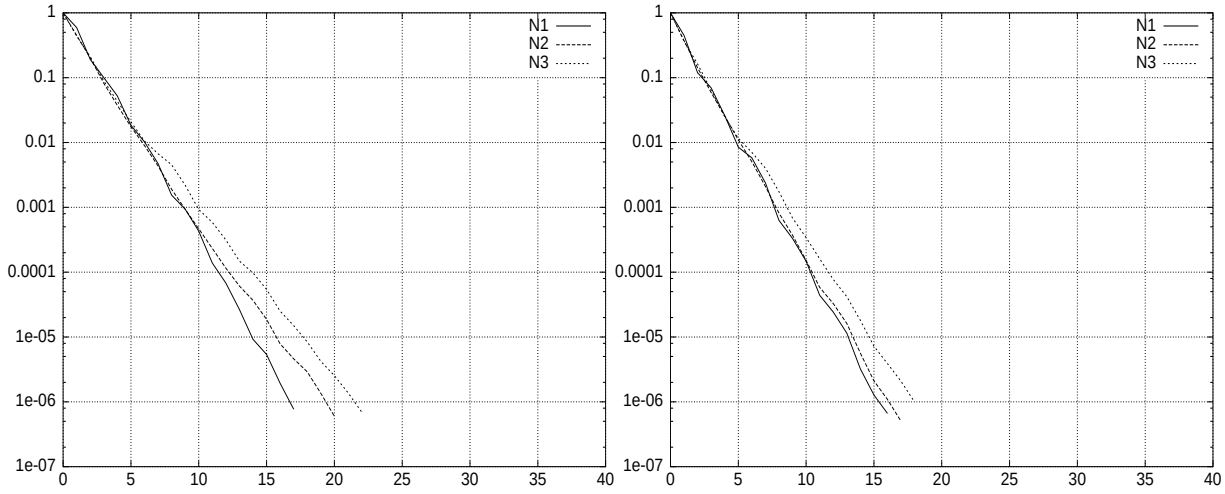


Figure 3.23: Transonic flow test case
Additive ideal two-grid algorithm with residual filtering
Left : $\nu_{add} = 3$ - Right : $\nu_{add} = 4$

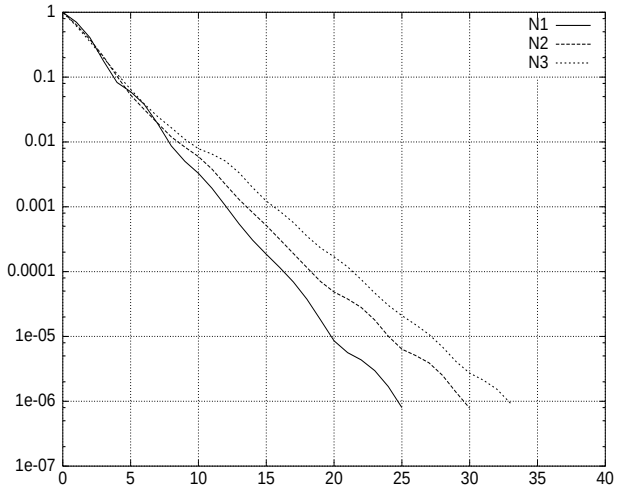


Figure 3.24: Transonic flow test case: additive ideal two-grid algorithm
Residual and correction filtering, $\nu_{add} = 2$

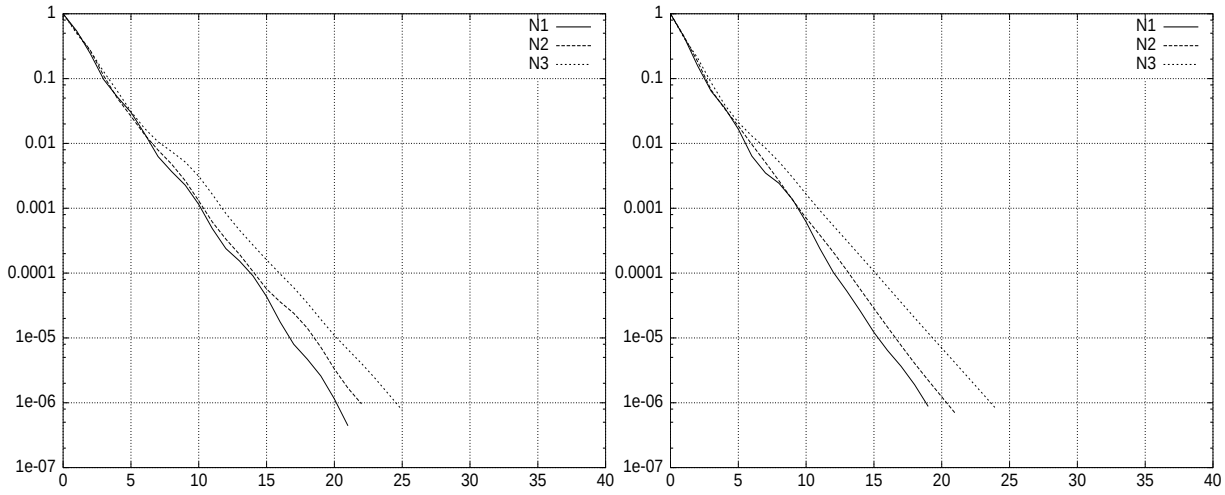


Figure 3.25: Transonic flow test case
Additive ideal two-grid algorithm with residual and correction filtering
Left : $\nu_{add} = 3$ - Right : $\nu_{add} = 4$

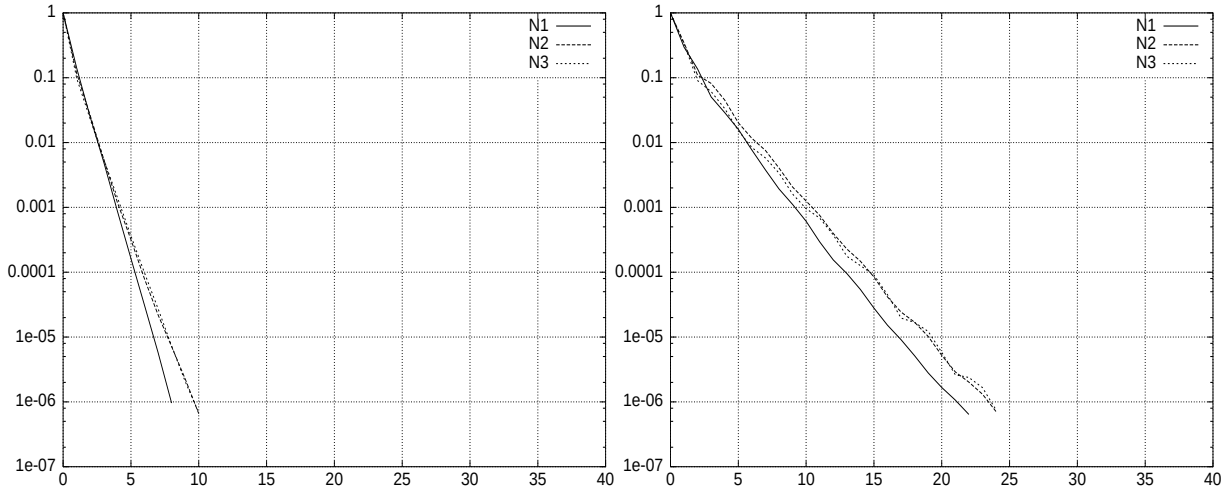


Figure 3.26: Subsonic flow test case
Left : multiplicative ideal two-grid algorithm
Right : additive ideal two-grid algorithm with residual filtering, $\nu_{add} = 2$

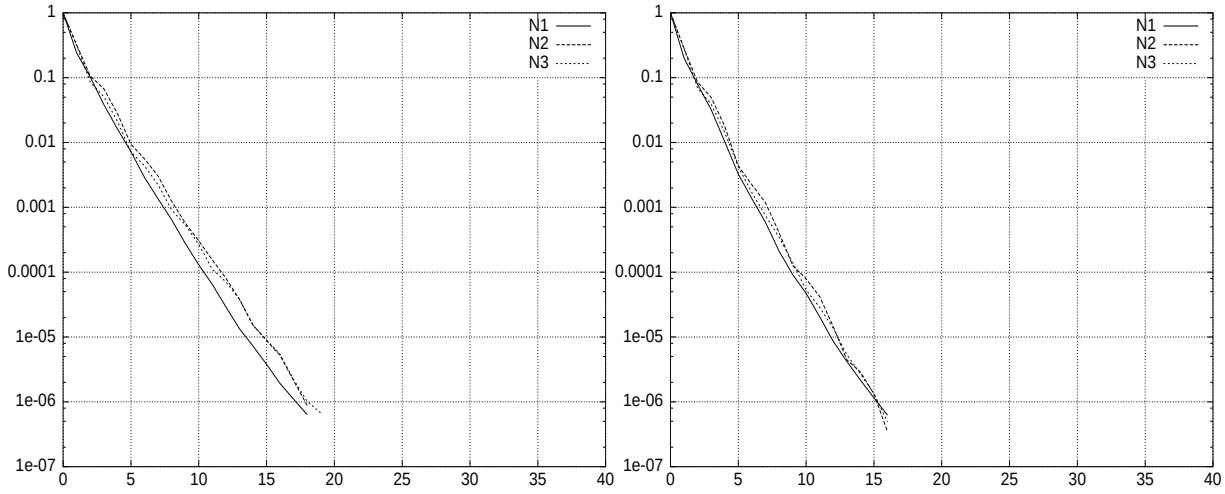


Figure 3.27: Subsonic flow test case
Additive ideal two-grid algorithm with residual filtering
Left : $\nu_{add} = 3$ - Right : $\nu_{add} = 4$

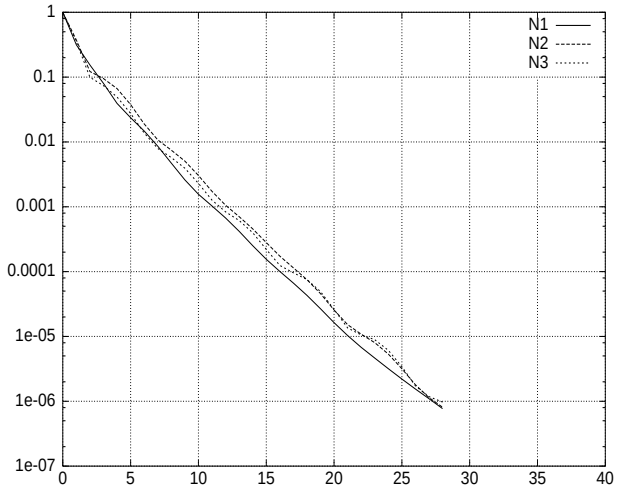


Figure 3.28: Subsonic flow test case: additive ideal two-grid algorithm
Residual and correction filtering, $\nu_{add} = 2$

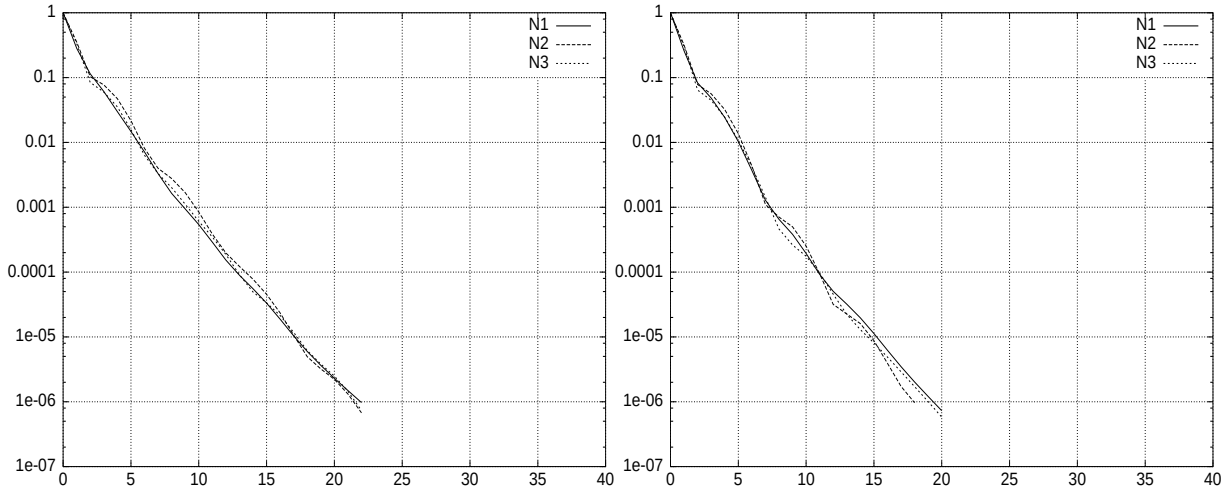


Figure 3.29: Subsonic flow test case
Additive ideal two-grid algorithm with residual and correction filtering
Left : $\nu_{add} = 3$ - Right : $\nu_{add} = 4$

3.2.4.4.1 Transonic flow test case. Steady contour lines of the Mach number for this test case are shown on figure 3.30 (left figure). Performance results are given in tables 3.11 and 3.12. Two strategies have been considered:

- at each time step, the linear system (1.84) is approximately solved based on a threshold which has been fixed to $\varepsilon = 10^{-1}$. The corresponding results are given in table 3.11;
- at each time step, the linear system (1.84) is approximately solved using a fixed number of multigrid cycles $N_{c_{max}} = 4$. The corresponding results are given in table 3.12.

We note that increasing the computational effort for solving the underlying linear systems, either by reducing the linear threshold ε by one or more orders of magnitude or, by imposing a greater number multigrid cycles $N_{c_{max}}$, has not resulted in a lower number of pseudo-time steps for the convergence to the steady state solution. Therefore, the situations selected here are somewhat optimal (as far as the total simulation time is concerned).

3.2.4.4.2 Subsonic flow test case. Steady contour lines of the Mach number for this test case are shown in figure 3.30 (right figure). The same resolution strategies as those considered for the transonic test case have been adopted here. We simply note that for the second strategy, we have used a fixed number of multigrid cycles $N_{c_{max}} = 3$ (instead of $N_{c_{max}} = 4$ for the transonic test case). Performance results are given in table 3.14 and table 3.15.

3.2.4.4.3 Comments. This second set of results calls for the following comments:

- the additive algorithm, as implemented here, often behaves better (in terms of the total simulation time) than the multiplicative one. This fact is especially true for the simulations using $N_p \geq 4$ and is mainly due to our choice of limiting the application of the additive formulation to the two coarsest grid levels of the multigrid hierarchy. This implementation results in a slight degradation of the numerical efficiency for the overall hybrid multiplicative/additive algorithm compared to the standard multiplicative algorithm. Recall that on the finest levels, the parallel efficiency of the smoothing steps is generally high; therefore, we cannot expect a notable gain by switching from the multiplicative algorithm to the additive one on these levels;
- the CPU utilization is always higher for the hybrid multiplicative/additive algorithm (as soon as $N_p \geq 4$). Clearly, this gain is due to the reduction of the communication cost on the two coarsest grid levels, according to our choice of implementation. In some cases we observe a 4% difference in the CPU utilization between the MUL and ADD strategies while the degradation of parallel efficiency for the multiplicative algorithm reaches 20%. We can reasonably expect higher improvements with an application of the additive algorithm to more than the two coarsest levels, even though, as mentioned in the previous point, it is important to preserve the hybrid multiplicative/additive nature of the algorithm in order to obtain a good compromise between parallel efficiency and numerical efficiency;
- the improvement of the parallel speed-up is never in accordance with the reduction of the communication cost. This behavior is unfortunately inherent to the partitioning strategy adopted here which allows for a one element overlap at submesh interfaces. This translates into redundant arithmetic operations that are taken into account in the simulation times for $N_p \geq 2$. In [84] an analysis of the influence of the partitioning strategy for three-dimensional implicit (monogrid) calculations shows that these redundant arithmetic operations play a major role in the degradation of the speed-up;
- finally, to conclude this discussion, we give in table 3.13 and 3.16 the results obtained using the monogrid algorithm, for both test cases, using the Jacobi method as an iterator for the approximate solution of the implicit system at each time step, according to a threshold which has been fixed to $\varepsilon = 10^{-1}$. From these measures we can see that the gain using the additive multigrid algorithm is equal to 3.8 for the transonic test case and 4.5 for the subsonic one.

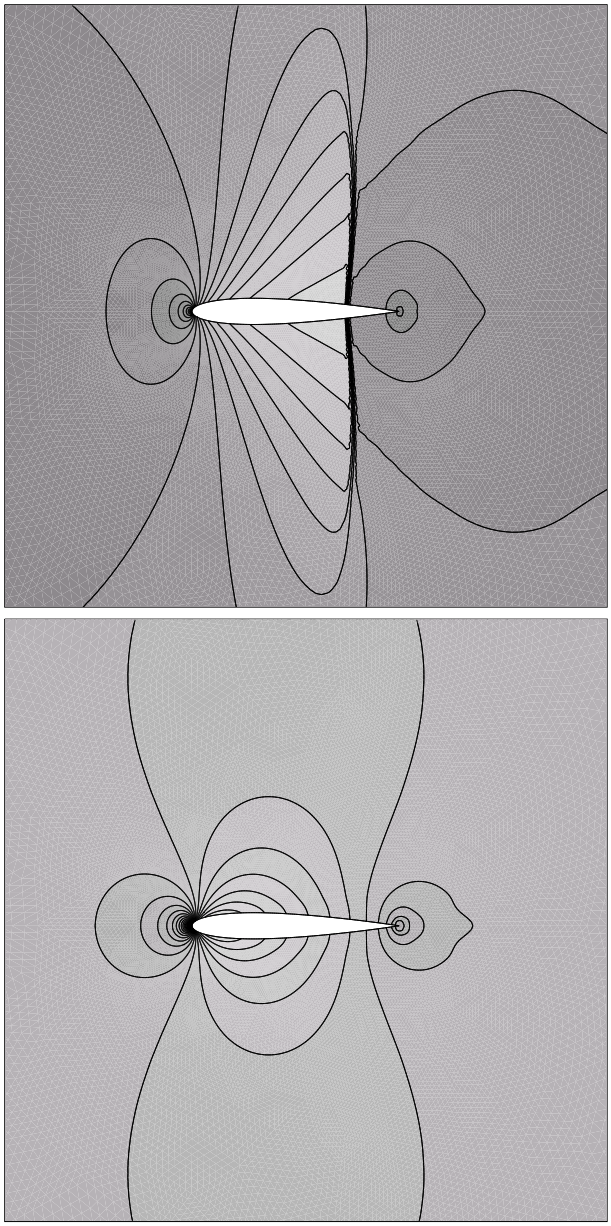


Figure 3.30: External flow around the `NACA0012` airfoil: steady contour lines of the Mach number
Top: transonic flow - Bottom: subsonic flow

Table 3.11: Multiplicative and additive multigrid algorithms
Transonic test case (mesh N3) : $\varepsilon = 10^{-1}$ and $N_{c_{max}} = 100$

ALGO	# i_t	N_p	Total time	CPU time	% CPU	$S(N_p)$
MUL	98	2	1738 sec	1672 sec	96.0	1.0
	98	4	982 sec	886 sec	90.0	1.8
	98	8	564 sec	494 sec	87.5	3.1
	98	12	491 sec	389 sec	79.5	3.5
ADD	100	2	1806 sec	1754 sec	97.0	1.0
	99	4	983 sec	924 sec	94.0	1.8
	100	8	590 sec	535 sec	90.5	3.1
	101	12	490 sec	397 sec	83.5	3.7

Table 3.12: Multiplicative and additive multigrid algorithms
Transonic test case (mesh N3) : $\varepsilon = 10^{-10}$ and $N_{c_{max}} = 4$

ALGO	# i_t	N_p	Total time	CPU time	% CPU	$S(N_p)$
MUL	98	2	1871 sec	1753 sec	93.5	1.0
	98	4	1045 sec	948 sec	90.5	1.8
	98	8	618 sec	530 sec	86.0	3.0
	98	12	539 sec	433 sec	80.5	3.5
ADD	100	2	1810 sec	1781 sec	98.5	1.0
	99	4	1020 sec	946 sec	93.0	1.8
	100	8	608 sec	543 sec	90.0	3.0
	100	12	510 sec	426 sec	83.5	3.6

Table 3.13: Transonic test case (mesh N3) : monogrid algorithm with $\varepsilon = 10^{-1}$

# i_t	N_p	Total time	CPU time	% CPU
104	12	1912 sec	1771 sec	92.5

Table 3.14: Multiplicative and additive multigrid algorithms
Subsonic test case (mesh N3) : $\varepsilon = 10^{-1}$ and $N_{c_{max}} = 100$

ALGO	# i_t	N_p	Total time	CPU time	% CPU	$S(N_p)$
MUL	62	2	1249 sec	1189 sec	95.0	1.0
	62	4	699 sec	640 sec	91.5	1.8
	62	8	416 sec	373 sec	89.5	3.0
	62	12	346 sec	279 sec	80.5	3.6
ADD	62	2	1257 sec	1215 sec	96.5	1.0
	62	4	691 sec	654 sec	94.5	1.8
	62	8	414 sec	376 sec	91.0	3.0
	62	12	333 sec	279 sec	84.0	3.7

Table 3.15: Multiplicative and additive multigrid algorithms
Subsonic test case (mesh N3) : $\varepsilon = 10^{-10}$ and $N_{c_{max}} = 3$

ALGO	# i_t	N_p	Total time	CPU time	% CPU	$S(N_p)$
MUL	62	2	913 sec	867 sec	95.0	1.0
	62	4	546 sec	503 sec	92.0	1.7
	62	8	324 sec	282 sec	87.0	2.8
	62	12	268 sec	215 sec	80.0	3.4
ADD	62	2	949 sec	906 sec	95.5	1.0
	62	4	532 sec	492 sec	92.5	1.8
	62	8	317 sec	284 sec	89.5	3.0
	62	12	251 sec	210 sec	83.5	3.8

Table 3.16: Subsonic test case (mesh N3) : monogrid algorithm with $\varepsilon = 10^{-1}$

# i_t	N_p	Total time	CPU time	% CPU
61	12	1138 sec	997 sec	87.5

3.2.5 Conclusion and future work

In this section, we have studied two parallel multigrid formulations for the acceleration of compressible steady flow calculations on unstructured meshes. In particular, we have performed a detailed evaluation of an additive formulation based on a residual/correction filtering technique, extending an idea originally proposed by Chan and Tuminaro[30]. This technique has been studied as a mean for reducing the communication overheads of coarse grid operations in parallel multigrid algorithms. A concrete implementation has been proposed where the additive formulation is applied to the two coarsest levels of the multigrid hierarchy resulting in a hybrid multiplicative/additive multigrid algorithm. Even with this preliminary implementation, interesting results have been obtained demonstrating that the hybrid algorithm is always more efficient than the classical multiplicative multigrid algorithm.

Concerning future work, the main objective is clearly the extension of the hybrid multiplicative/additive multigrid algorithm to more complex physical models (laminar and turbulent Navier-Stokes equations for compressible flows). In addition several directions are currently investigated:

- extension of the effective utilization of the additive formulation to more than two grid levels. From the numerical efficiency point of view, the results reported here have shown that the hybrid multiplicative/additive multigrid algorithm, where the additive formulation is applied to the two coarsest grid levels, is competitive with the classical multiplicative multigrid algorithm. Clearly, there is room for improving the overall efficiency of the hybrid multiplicative/additive MG algorithm as far as the loss of numerical efficiency induced by the additive formulation is compensated by the gain in parallel efficiency;
- the multiplicative and the hybrid multiplicative/additive multigrid algorithms could be used as a preconditioner to a Krylov type acceleration method. A smoothing analysis such as the one given in [144] would be useful in this context;
- the smoother adopted in this study is one of the simplest. It is also a parallel one and that was the main motivation for using it here. Assessing the behavior of the hybrid multiplicative/additive multigrid algorithm using more efficient smoother is mandatory for a consolidation of the present work.

3.3 Non-linear multi-mesh multigrid methods

The multigrid methods considered in sections 3.1 and 3.2 make use of a volume agglomeration principle for the automatic construction of a hierarchy of coarse grid levels. The main advantage of this approach certainly relies in the fact that the only input to the method is the discretization grid which is assumed to be the support of the targeted flow solution. In addition, the multigrid by volume agglomeration method is particularly well suited to the mixed element/volume formulation on unstructured triangular or tetrahedral meshes used in sections 3.1 and 3.2 for the discretization of the Euler and Navier-Stokes equations. However, the multigrid by volume agglomeration method also has two mains drawbacks: on one hand, its implementation is relatively tricky especially when it is used as an acceleration method for the resolution of linear systems (i.e. linear multigrid method) and, in all cases, requires a good knowledge of the implementation details of the underlying space discretization method; on the other hand, its numerical efficiency depends on the approximation method used for the construction of the coarse grid operators which may call for some approximation since it is not possible to apply a finite element type formulation on the coarse grid levels. An illustration of the second point in the context of the mixed element/volume formulation described in section 1.2.1 of chapter 1, stands in the treatment of diffusive terms on the coarse grid levels (see also Carré[23] for more details). Clearly, a multi-mesh strategy for which the same type of discretization mesh is used whether it is the finest grid level or any of the coarse grid levels, allows to apply the same space discretization method on each level of the multigrid hierarchy. Concerning the first point, one possible strategy to simplify the implementation of a multigrid method is to adopt a non-linear (FAS) multigrid method instead of a linear multigrid method such as the one considered in sections 3.1 and 3.2.

In this section, we describe our contributions concerning the development of non-linear multi-mesh multigrid algorithms for the acceleration of three-dimensional compressible flows on unstructured tetrahedral meshes. The corresponding work was undertaken in the framework of the BRITE EURAM 3/ IDeMAS project (from December 1st 1997 to June 30th 2001).

The goal of the IDeMAS project ("Industrial Demonstration of accurate and Efficient Multidimensional and multigrid Algorithms for aerodynamic Simulation on unstructured grids") was the design of a new generation CFD solver based on recent developments in the following fields:

- multidimensional upwind high resolution schemes on tetrahedral meshes for the discretization of Euler and Navier-Stokes (laminar and turbulent) equations modeling compressible flows[111]-[35];
- implicit time integration based on Newton's method with first order analytical Jacobians or second order numerical Jacobians, combined with preconditioned Krylov solvers[71];
- non-linear multi-mesh multigrid algorithms on tetrahedral meshes[26];
- solution adaptivity for tetrahedral meshes.

The partners of this project were: von Karman Institute (VKI), Ecole Polytechnique Fédérale de Lausanne (EPFL), Institut National de Recherche en Informatique et Automatique (INRIA), Centro di Ricerca, Sviluppo e Studi Superiori in Sardegna (CRS4), Dassault Aviation, Daimler-Benz Aerospace (DASA) and Alenia Aerospazio.

The multigrid strategy adopted in IDeMAS is of the non-linear type i.e. the so-called Full Approximation Scheme (FAS) method. In this context, the objective was to optimize the time advancing strategy already existing in the starting-point monogrid solver from the point of view of the number of operations required to reach a specified solution state (for both steady or unsteady flows). In the multi-mesh approach, several independent and possibly not-nested finite element meshes are considered as input data to the multigrid algorithms. An important consequence of this choice is that the inter-grid transfer operators must be general and, as a matter of fact, cannot be trivially constructed. In this section, we focus on those multigrid aspects. In doing so, we limit our presentation to the calculation of inviscid compressible flows modeled by the three-dimensional Euler equations. The rest of this section is organized as follows: in subsection 3.3.1, we summarize the main characteristics of the starting-point flow solver and we outline the monogrid formulation of the problem under consideration; in subsection 3.3.2, we describe the algorithmic aspects of the non-linear multi-mesh multigrid algorithms that are at the heart of this study; subsection 3.3.5 is concerned with parallel computing aspects; finally, in subsection

3.3.6, we present numerical results that aim at assessing the numerical and parallel efficiencies of the developed multigrid algorithms.

3.3.1 Monogrid formulation of the problem

In IDeMAS, the compressible flow solver that we considered as a starting point for our study was given by the `THOR` CFD package[140] whose main features are summarized below:

- multidimensional upwind high resolution schemes on unstructured tetrahedral meshes for the discretization of the system of Euler equations[111]-[35]. An introduction to these numerical schemes is given in section 1.2.2 of chapter 1. We recall that an important characteristic of these schemes is that they rely of a selective distribution of an elementary flux computed on an element (tetrahedron) to the vertices of this element (i.e. they are compact schemes);
- concerning time integration to steady state, both explicit (Runge-Kutta method) and implicit (backward Euler combined with a Newton method) are available. In the latter case, the sparse and non-symmetric linear systems resulting from the linearization of the Euler equations are iteratively solved using preconditioned Krylov methods such as GMRES[124] combined with a block incomplete factorization method (BILU). The Krylov methods and preconditioners are taken from the `Aztec` library[71] developed Sandia National Laboratories;
- the SPMD parallelization of `THOR` calls for a classical strategy that combines an appropriate partitioning of the underlying tetrahedral mesh and a message passing programming model.

Let W denotes the vector of state variables (for instance, the conservative variables ρ, $\rho\vec{V}$, E where ρ is the fluid density and E is the total energy per unit of volume). We suppose that the discretization of the system of Euler equations has resulted in the following set of ordinary differential equations :

$$\frac{dW_i}{dt} + \Psi(W)_i = 0 \quad , \quad i = 1, \cdots, N_V \tag{3.36}$$

In eq. (3.36), N_V denotes the number of mesh vertices and $\Psi(W) = -R(W)$ where $R(W)$ stands for the residual vector of eq. (1.77) in section 1.2.2 of chapter 1. This new notation of the residual vector has been adopted to avoid any confusion with the notations adopted below for the description of the multigrid algorithms. Let $W^n = W(x, n\Delta t)$ be the solution at time $t^n = n\Delta t$. In our study, only steady-state problems are considered. Therefore, eq. (3.36 is solved iteratively until convergence of the temporal term to zero. In the following subsection, we introduce multigrid algorithms for the acceleration of the convergence of this iterative process. This is done in the framework of an explicit Runge-Kutta method nevertheless the extension of the algorithms to the case of a linearized implicit scheme is straightforward.

3.3.2 Non-linear multigrid acceleration

3.3.2.1 Notations

For the purpose of the description of the multigrid algorithms, we introduce the following notations:

- $\{G_k\}_{1 \leq k \leq K}$ denotes a sequence of possibly not-nested tetrahedral meshes. G_1 stands for the finest grid while G_K is the coarsest one.
- N_k is the number of mesh vertices of grid G_k.
- $\mathcal{N}_k$ is the list of mesh vertices of grid G_k.
- $W_k = \{(W_k)_{i,j}\} \quad , \quad 1 \leq i \leq N_k \quad , \quad 1 \leq j \leq N_{sv}$ is the global state vector associated with grid G_k where N_{sv} is the number of state variables which depends on the space dimension and on the underlying flow model (i.e. Euler, laminar Navier-Stokes, turbulent Navier-Stokes). In this study, $N_{sv} = 5$.

- $\Psi_k(W)$ is the global numerical flux vector associated with grid G_k.

We will call, and denote by L_k, *smoothing operator* associated with grid G_k, the following application (not necessarily linear):

$$(W_k, S_k) \stackrel{L_k}{\longmapsto} \overline{W}_k$$

where $\overline{W}_k$ denotes an approximate solution to the system :

$$\Psi_k(W_k) = S_k$$

S_k being a source term that will be defined later. In our context, one application of the smoothing operator will consist in the realization of one iteration (pseudo-time step) of the time integration method used for time advancing eq. 3.36. We will call, and denote by R_k, *non-linear residual operator* associated with grid G_k, the following application:

$$(W_k, S_k) \stackrel{R_k}{\longmapsto} S_k - \Psi_k(W_k)$$

The definition of a multigrid cycle relies on the availability of inter-grid transfer operators. Let X_k, X_{k+1}, Y_k and Y_{k+1} denote the discrete functional spaces respectively associated to W_k, W_{k+1}, $\Psi_k(W_k)$ and $\Psi_{k+1}(W_{k+1})$. Three inter-grid transfer operators need to be defined:

- R_k^{k+1} (*restriction operator*) is an application from X_k to X_{k+1}. This operator is used for transferring the global state vector W_k from grid G_k (fine) to grid G_{k+1} (coarse);

- D_k^{k+1} (*distribution operator*) is an application from Y_k to Y_{k+1}. This operator is used for transferring the global residual vector $R_k = R_k(W_k, S_k)$ from grid G_k (fine) to grid G_{k+1} (coarse);

- P_{k+1}^k (*prolongation operator*) is an application from X_{k+1} to X_k. This operator is used for transferring the global correction vector C_{k+1} (to be defined later) from grid G_{k+1} (coarse) to grid G_k (fine).

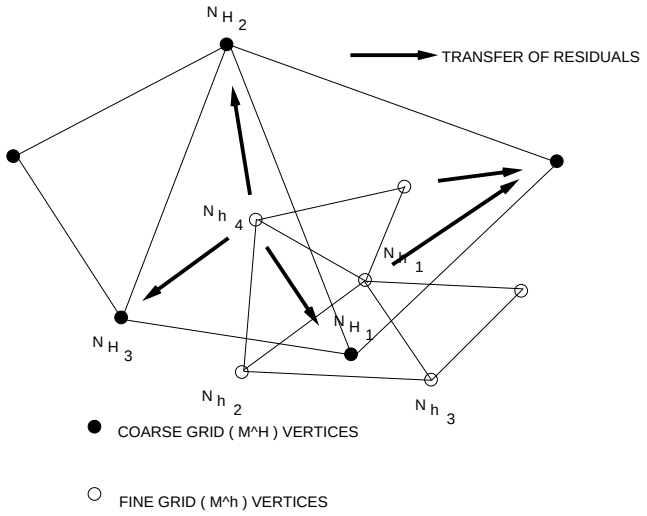


Figure 3.31: Transfer operators in 2D

3.3.2.2 Inter-grid transfer operators

The inter-grid transfer operators described here are independent of the nature of the problem under consideration. They are before all based on geometrical data. We respectively denote by τ_k and τ_{k+1} the tetrahedrizations corresponding to grids G_k and G_{k+1}. Since the underlying grids can be not-nested, the implementation of inter-grid transfer operators may require solving localization problems between successive grids of the multigrid hierarchy. We suppose that this preliminary step has been performed and we denote by $T_{k+1}(n_i^k) \in \tau_{k+1}$ (respectively $T_k(n_i^{k+1}) \in \tau_k$) the identifier of the tetrahedra of grid G_{k+1} (respectively G_k) that contains the vertex $n_i^k \in G_k$ (respectively $n_i^{k+1} \in G_{k+1}$). For each vertex n_i^{k+1} we also define the set:

$$\mathcal{S}_i^{k+1} = \{j \in \tau_k \text{ such that } n_i^{k+1} \text{ is a vertex of } T_{k+1}(n_j^k)\} \tag{3.37}$$

that is, $\mathcal{S}_i^{k+1}$ is the set of vertices of the fine grid G_k belonging to the finite element support of vertex n_i^{k+1} of G_{k+1}. Let $\varphi^{k+1}_{T_{k+1}(n_j^k),l}$ denotes the basis function associated to vertex l of $T_{k+1}(n_j^k)$.

3.3.2.2.1 Restriction operator. The result of the restriction of the global state vector W_k onto the coarse grid G_{k+1} is obtained by a *linear interpolation* based on the P1 finite element basis functions. Let n_l^k denote the vertices of element $T_k(n_i^{k+1}) \in \tau_k$. For each vertex $n_i^{k+1} \in \tau_{k+1}$ we obtain:

$$W_{k+1}(n_i^{k+1}) = R_k^{k+1}(W_k)(n_i^{k+1}) = \sum_{l=1,L} \varphi^k_{T_k(n_i^{k+1}),l}(n_i^{k+1}) W_k(n_l^k) \tag{3.38}$$

where $L = 3$ in 2D and $L = 4$ in 3D.

3.3.2.2.2 Distribution operator. The distribution of the global residual vector R_k onto the coarse grid G_{k+1} makes use of the elements of the set $\mathcal{S}_i^{k+1}$ (3.37) for each coarse grid vertices n_i^{k+1}. Recall that the Galerkin interpolation of a function $f(\vec{x})$ on an element T is given by:

$$f(\vec{x}) \mid_T = \sum_{l=1,L} \varphi_T(\vec{x}) f_l$$

where $f_l = f(\vec{x_l})$. In particular, the P1 finite element basis functions satisfy:

$$\sum_{l=1,L} \varphi_T(\vec{x}) = 1 \quad , \ \forall \vec{x}$$

The global value of the function $f(\vec{x})$ is obtained by gathering the elementary values $f(\vec{x}) \mid_T$. Using these remarks, a *linear distribution* scheme is used for the evaluation of the global residual vector R_{k+1}. For each vertex $n_i^{k+1} \in \tau_{k+1}$ we obtain (see figure 3.31):

$$R_{k+1}(n_i^{k+1}) = D_k^{k+1}(R_k)(n_i^{k+1}) = \sum_{j \in \mathcal{S}_i^{k+1}} \varphi^{k+1}_{T_{k+1}(n_j^k),i}(n_j^k) R_k(n_j^k) \tag{3.39}$$

3.3.2.2.3 Prolongation operator. The result of the prolongation of the global correction vector C_{k+1} onto the fine grid $\mathbf{G_k}$ is obtained by a *linear interpolation* based on the P1 finite element basis functions. Let n_l^{k+1} denote the vertices of element $T_{k+1}(n_i^k) \in \tau_{k+1}$. For each vertex $n_i^k \in \tau_k$ we obtain:

$$C_k(n_i^k) = P_{k+1}^k(C_{k+1})(n_i^k) = \sum_{l=1,L} \varphi^{k+1}_{T_{k+1}(n_i^k),l}(n_i^k) C_{k+1}(n_l^{k+1}) \tag{3.40}$$

where $L = 3$ in 2D and $L = 4$ in 3D.

3.3.2.3 The FAS V-cycle algorithm

The underlying strategy of the FAS multigrid method is to solve on the coarse grids the discretized non-linear equations including source terms coming from the finer grids. These coarse grid resolutions result in corrections that contribute to the construction of a new fine grid solution. The FAS algorithm relies on a defect correction type iteration. Let $\overline{W}_k$ denotes an approximate solution of the non-linear system $\Psi_k(W_k) = S_k$ defined on the fine grid G_k. The most common multigrid cycles are the V-cycle, the F-cycle and the W-cycle. Here we describe the simplest cycle i.e. the V-cycle.

The different steps of the algorithm are the following:

- *initialization* of the cycle : the input data is given by the fine grid global state vector W_1^{old} calculated at the previous cycle. On the fine grid G_1, the source term S_1 is set to 0.

- *downward phase* : $G_1 \rightarrow G_K$. The goal is to evaluate *coarse grid corrections* of the approximate solution $\overline{W}_1$ resulting from the smoothing of the input data W_1^{old}:

  - pre-smoothing step on the finest grid G_1 : $\overline{W}_1 = L_1(W_1^{old}, S_1)$
  - non-linear residual evaluation on the finest grid G_1 : $R_1 = R_1(\overline{W}_1, S_1)$
  - FOR $k = 2, \cdots, K$ DO
    - inter-grid transfers : $W_k = R_{k-1}^k(\overline{W}_{k-1})$ and $R_k = D_{k-1}^k(R_{k-1})$
    - calculation of the source term on grid G_k: $S_k = \Psi_k(W_k) + R_k$
    - smoothing step and non-linear residual update on grid G_k:

$$\overline{W}_k = L_k(W_k, S_k) \text{ and } R_k = R_k(\overline{W}_k, S_k)$$

    - calculation of a correction on grid G_k : $C_k = \overline{W}_k - W_k$
  - END FOR

- *upward phase* : $G_K \rightarrow G_1$. The goal is to evaluate a new iterate W_h^{new} using the prolongated coarse grid corrections :

  - FOR $k = K, \cdots, 2$ DO
    - solution update on grid G_{k-1} using the prolongated coarse grid correction :

$$\tilde{W}_{k-1} = \overline{W}_{k-1} + P_k^{k-1}(C_k)$$

    - smoothing step and solution update on grid G_{k-1}:

$$W_{k-1} = L_k(\tilde{W}_{k-1}, S_{k-1})$$

  - END FOR
  - $W_h^{new} = W_1$

3.3.2.4 FMG strategy

Multigrid principles often yield very efficient solvers for the algebraic systems of equations resulting from the discretizations of systems of PDEs. For linear elliptic problems, it is possible to show that a multigrid (MG) cycle is charactenrized by an h-independent convergence factor. This result and the fact that the number of operations per cycle is $O(N_V)$ operations where N_V is the number of grid points of the finest mesh, together imply the optimality of the MG cycle. In order to achieve a (fixed) error (or defect) reduction by a factor of ε, $O(N_V \log \varepsilon)$ operations are sufficient. The Full Multigrid (FMG) variant consists in initializing a series of MG cycles with an approximate solution which is computed on a coarser mesh and is assumed to be sufficiently close to the targeted solution (if the underlying mesh is coarse enough, this solution can be the result of an exact resolution of the problem defined on this mesh). This coarse mesh solution is then interpolated to the finer mesh of the multigrid hierarchy used for the MG iteration. Several MG cycles are performed using these two grid levels and the resulting solution is again prolongated to a finer mesh which is included in the multigrid hierarchy. This process is illustrated on figure 3.32 for a MG iteration based on a V-cycle. Under natural assumptions, FMG provides approximations with discretization accuracy. Together with the fact that the number of operations is $O(N_V)$, this is the reason for the optimality of FMG: discretization accuracy is achieved in $O(N_V)$ operations.

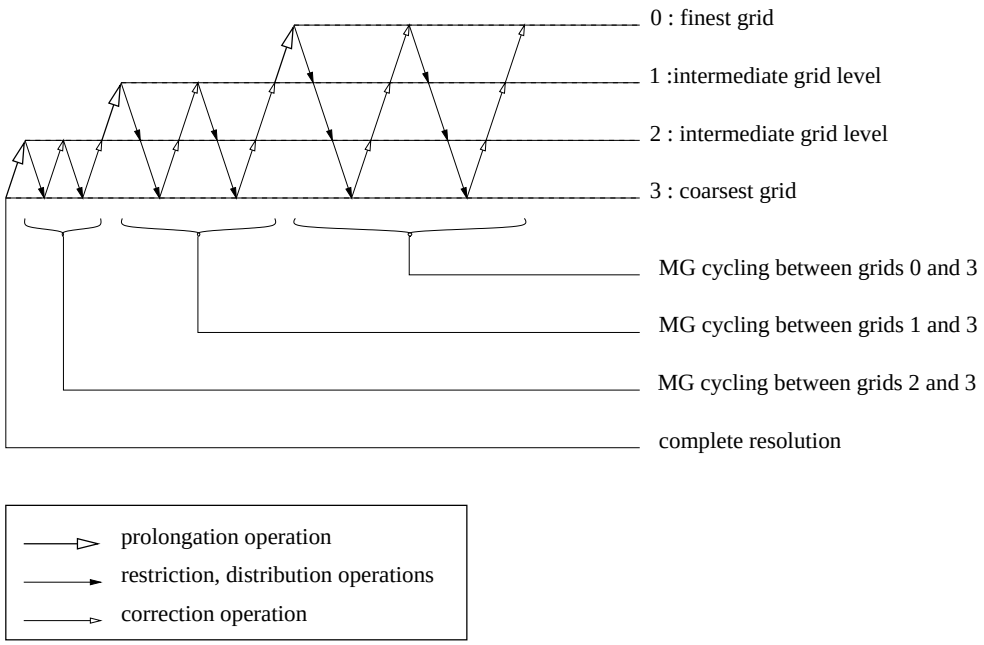


Figure 3.32: Standard FMG strategy

3.3.2.4.1 Standard FMG strategy: outline of the theory. As mentioned above, the first step of the FMG strategy consists in computing a fully converged solution of the problem on the coarsest grid level. The second step is to prolongate this coarse grid solution on a finer grid. By this wemean, one obtains an appropriate initial guess to a classical MG cycle acting on these two grid levels. Because this initial guess is obtained from a fully converged solution, one can determine a stopping criterion for the subsequent MG cycling in order to insure that the error of the fine grid solution is smaller than the order of accuracy of the spatial discretization scheme. In this paragraph, we recall some theoretical aspects underlying the evaluation of this stopping criterion for the MG cycling, in each phase of the FMG strategy. For this purpose, we make use of a series of finite element subspaces $\mathcal{H}_1, \cdots, \mathcal{H}_N$ defining a sequence of not-nested meshes included in the Hilbert space $(\mathcal{H}, \| \, . \, \|)$.

Lemma 1 *Let Π_k be an operator from $\mathcal{H}$ to $\mathcal{H}_k$, u and u_k the solutions of a variational problem in $\mathcal{H}$ and $\mathcal{H}_k$ respectively. If the following estimations are verified (where p is the consistency order of the discretization, q the interpolation (prolongation) order, P_{k+1}^k the prolongation operator from grid G_{k+1} to grid G_k and h_k the discretization step on grid level k):*

$$\| \, \Pi_k(u) - u_k \, \| = O(h_k^p) \tag{3.41}$$
$$\| \, \Pi \, \| = O(1) \tag{3.42}$$
$$\| \, P_{k+1}^k \, \| = O(1) \tag{3.43}$$
$$\| \, P_{k+1}^k \left(\Pi_{k+1}(u) \right) - \Pi_k(u) \, \| = O(h_k^q) \tag{3.44}$$
$$\tag{3.45}$$

then the relative discretization error of u_{k+1} compared to u_k is given by:

$$\| \, P_{k+1}^k(u_{k+1}) - u_k \, \| \leq C_1 h_k^{\min(p,q)} \tag{3.46}$$

The proof of this lemma can be found in [102] and [22]. From now on, we assume that the interpolation order q is greater than the consistency order p of the spatial discretization. This means that eq. (3.46) is written:

$$\| P_{k+1}^{k}(u_{k+1}) - u_k \| \leq C_1 h_k^p \tag{3.47}$$

Definition 1 *Let s_k be the reduction factor of a MG cycle on grid level G_k, $u_k^{(n)}$ the approximation of the discrete solution after the n-th MG cycle, then:*

$$\| u_k - u_k^{(n+1)} \| \leq s_k \| u_k - u_k^{(n)} \| \qquad \forall k \in [1, N-1] \;\; , \;\; s_k \leq s < 1 \tag{3.48}$$

where s is the reduction factor of the MG process independent of the grid level (u_k is the initial guess).

Lemma 2

$$\| u_k - u_k^{(n)} \| \leq C_3 C_1 h_k^p \tag{3.49}$$

where:

$$C_3 = \frac{s^n}{1 - C_2 s^n} \quad \textit{with} \quad C_2 = \sup_k \left(\frac{h_{k+1}}{h_k} \right)^p > 1 \tag{3.50}$$

The proof of this lemma is also given in [102]. Since $C_2 > 1$, we obviously have:

$$\| u_k - u_{k+1}^{(n)} \| \leq \frac{C_2 s^n}{1 - C_2 s^n} C_1 h_k^p \tag{3.51}$$

In order to reach the discretization accuracy, one must have:

$$\| \Pi_k(u) - u_k^{(n)} \| = O(h_k^p) \tag{3.52}$$

The left-hand side of eq. (3.52) can be bounded as:

$$\| \Pi_k(u) - u_k^{(n)} \| \leq \| \Pi_k(u) - u_k \| + \| u_k - u_k^{(n)} \| \tag{3.53}$$

From eq. (3.41) we have that the first term of the right-hand side of (3.53) is $O(h_k^p)$. Now, if we can insure that:

$$\| u_k - u_k^{(n)} \| = O(h_k^p) \tag{3.54}$$

then condition (3.52) will be fulfilled. In order to satisfy condition (3.54) with the help of eq. (3.51) and assuming that the coefficient C_1 is small enough, Morano[102] proposed to impose the following condition:

$$\frac{C_2 s^n}{1 - C_2 s^n} \leq 1 \tag{3.55}$$

This leads to:

$$C_2 s^n \leq \frac{1}{2} \tag{3.56}$$

For a second order accurate discretization scheme ($p = 2$) and assuming that $C_2 \approx 2$, one obtains:

$$s^n \leq \frac{1}{8} \tag{3.57}$$

We assume that if the iterative error is reduced by a factor K then the residual is also reduced by the same factor. Form this, we can apply criterion (3.57) to the residual for deciding when to stop the MG cycling.

In practice, the initial residual $R_k^{(0)}$ on grid level G_k is defined using the prolongated solution from grid level G_{k+1} and the MG cycling is stopped when the current residual $R_k^{(n)}$ is such that:

$$\frac{R_k^{(n)}}{R_k^{(0)}} < \frac{1}{10} \tag{3.58}$$

3.3.2.4.2 Influence of the prolongation operator. In order to insure that criterion (3.58) leads to an accurate solution, we must estimate the initial residual $R_k^{(0)}$. If the result of the application of the prolongation operator to the coarse grid solution is polluted by high frequency error terms then the initial residual $R_k^{(0)}$ will have a high value. Because a MG smoother is by definition an efficient damping method for the upper part of the iterative error spectrum, it will rapidly remove these high frequency terms with only minor changes on the initial solution. The main consequence of this process is a rapidly decreasing residual in such a way that criterion (3.58) on the residual is satisfied whereas the solution has not been converged sufficiently. Carré[22] has illustrated this behavior for a simple 1D Poisson problem, This is shown on figure (3.33) for a linear interpolation method from a 3 nodes coarse grid to a 5 nodes fine grid. It is seen that one node over two has a null error and the other one has an important error. This leads to a high frequency error and, in some sense, to an artificially high value of the initial residual. The remedy to this problem is to use a high order prolongation operator based on a quadratic or a cubic interpolation method. By this mean, the artificial high frequency error term is notably reduced.

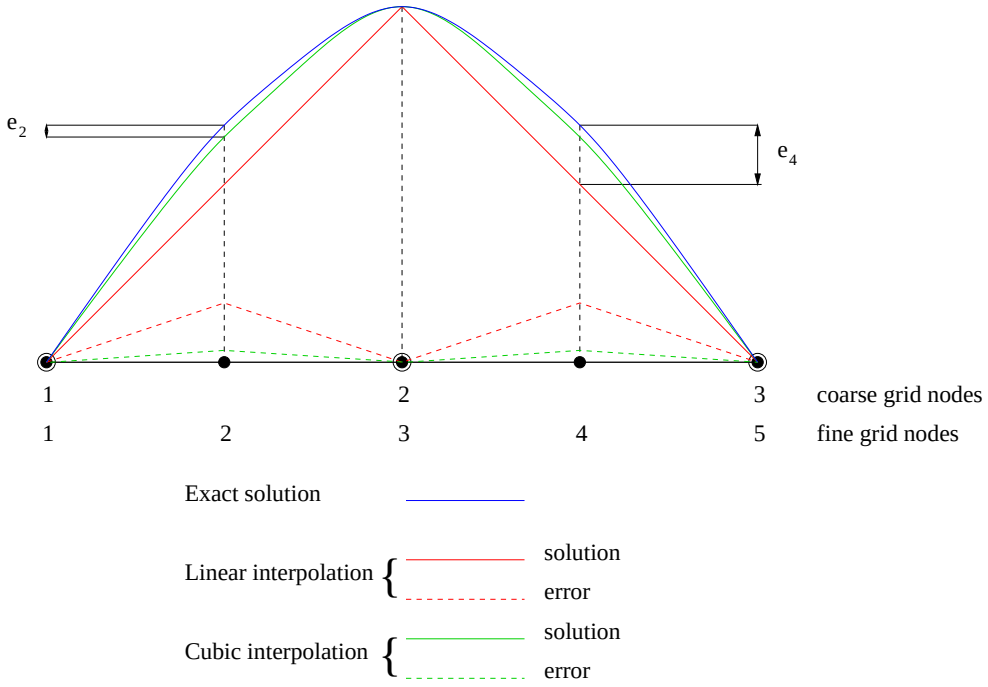


Figure 3.33: Impact of the prolongation operator on the initial solution

3.3.2.4.3 Failing FMG computations. Despite the use of high order prolongation operator, failing FMG computations have been reported by Carré and Dervieux [24] in the case of convection dominated viscous flow problems. In their study, the authors use a multigrid by agglomeration method for the acceleration of two-dimensional Navier-Stokes calculations on unstructured triangular meshes. The 2D viscous flow around a `NACA0012` airfoil, characterized by a free-stream Mach number equal to 0.8, an angle of attack which has been fixed to 10° and a Reynolds number equal to 73, has been computed with a standard FMG strategy. With these parameters, the FMG computation gives accurate results. Deviation of the pressure lift and drag coefficients is less than 10^{-5} compared to values obtained with a FAS computation with zero machine convergence. However, when the Reynolds number is increased to 500 (everything else being unchanged), the standard FMG is not able to produce an accurate approximation of the fully converged discrete solution. Final values of the pressure lift and drag coefficients present a large deviation with respect to the fully converged values. According to the authors, the initial solution on the fine mesh τ_k obtained by prolongating the solution of the coarse mesh τ_{k+1} is not adapted to τ_k. In other words, we have that $\| P_{k+1}^k(u_{k+1}^{(n)}) - u_k \| \neq O(h_k^p)$ because one pf the assumptions (3.41) to

(3.44) is not satisfied. Two strategies can be used to cure this problem: (1) the number of MG cycles can be increased arbitrarily or, (2) the quality of the solution obatined on τ_k can be estimated (since u_k is not avalable). In [24], the authors adopt the second strategy and propose a method to detect and characterize the problem, resulting in the so-called *controlled* (or *residual monitored*) FMG strategy. This method essentially consists in comparing the initial residuals $R_{k+1}^{(0)}$ (after prolongation of the solution from grid level $k+2$ to grid level $k+1$) and $R_k^{(0)}$ (after prolongation of the solution from grid level $k+1$ to grid level k). For a second order accurate discretization scheme ($p=2$) and if the ratio $\dfrac{h_{k+1}}{h_k}=2$, one should have:

$$\frac{R_k^{(0)}}{R_{k+1}^{(0)}} < \frac{1}{4} \tag{3.59}$$

This comparison presents a difficulty that arises from the fact that the finite element approximation is not node-wise consistent except on structured meshes. This behavior is illustrated below using a model problem. Indeed, let us consider the following 2D Poisson problem with Dirichlet boundary conditions:

$$\forall (x,y)\in\Omega \quad : \quad -\triangle u(x,y) = f(x,y) \quad \text{with} \quad u=0 \quad \text{on} \quad \partial\Omega \tag{3.60}$$

Let ϕ_i be the nodal basis function associated to node s_i of a triangulation τ of Ω containing a total of N_n nodes. The variational finite element formulation of the Poisson problem (3.60) is:

$$\forall i\in[1,N_n] \quad : \quad \sum_{j=1}^{N_n} u_j \int_\Omega \vec{\nabla}\phi_j.\vec{\nabla}\phi_i d\Omega = \int_\Omega f\phi_i d\Omega \tag{3.61}$$

Let us assume that the domain Ω is discretized using the mesh partially depicted on figure 3.34 and let us consider in more details the discretization of the Poisson equation at node 5. In doing so, $\phi_{i,k}$ denotes the P1 basis function associated to node s_i and defined on triangle t_k ($\phi_{i,k}(s_i)=1$ and 0 on the other nodes). The discretized form of the Poisson equation at node 5 involves the triangles τ_1, τ_2, τ_3 and τ_4 and is written:

$$\begin{aligned} S\Big[u_1\left(\vec{\nabla}\phi_{1,4}.\vec{\nabla}\phi_{5,4}+\vec{\nabla}\phi_{1,1}.\vec{\nabla}\phi_{5,1}\right) &+ u_2\left(\vec{\nabla}\phi_{2,1}.\vec{\nabla}\phi_{5,1}+\vec{\nabla}\phi_{2,2}.\vec{\nabla}\phi_{5,2}\right) \\ &+ u_3\left(\vec{\nabla}\phi_{3,2}.\vec{\nabla}\phi_{5,2}+\vec{\nabla}\phi_{3,3}.\vec{\nabla}\phi_{5,3}\right) \\ &+ u_4\left(\vec{\nabla}\phi_{4,3}.\vec{\nabla}\phi_{5,3}+\vec{\nabla}\phi_{4,4}.\vec{\nabla}\phi_{5,4}\right) + \\ u_5\left(\vec{\nabla}\phi_{5,1}.\vec{\nabla}\phi_{5,1}+\vec{\nabla}\phi_{5,2}.\vec{\nabla}\phi_{5,2}+\vec{\nabla}\phi_{5,3}.\vec{\nabla}\phi_{5,3}+\vec{\nabla}\phi_{5,4}.\vec{\nabla}\phi_{5,4}\right)\Big] &= \sum_{i=1}^{4} \int_{\tau_i} f\phi_{5,i} s\tau_i \end{aligned} \tag{3.62}$$

where $S=\dfrac{h^2}{2}$ is the area of one triangle. Let us call a_i the coefficient of u_i. We have:

$$\begin{cases} a_1 &= \vec{\nabla}\phi_{1,4}.\vec{\nabla}\phi_{5,4}+\vec{\nabla}\phi_{1,1}.\vec{\nabla}\phi_{5,1} \\ a_2 &= \vec{\nabla}\phi_{2,1}.\vec{\nabla}\phi_{5,1}+\vec{\nabla}\phi_{2,2}.\vec{\nabla}\phi_{5,2} \\ a_3 &= \vec{\nabla}\phi_{3,2}.\vec{\nabla}\phi_{5,2}+\vec{\nabla}\phi_{3,3}.\vec{\nabla}\phi_{5,3} \\ a_4 &= \vec{\nabla}\phi_{4,3}.\vec{\nabla}\phi_{5,3}+\vec{\nabla}\phi_{4,4}.\vec{\nabla}\phi_{5,4} \\ a_5 &= \vec{\nabla}\phi_{5,1}.\vec{\nabla}\phi_{5,1}+\vec{\nabla}\phi_{5,2}.\vec{\nabla}\phi_{5,2}+\vec{\nabla}\phi_{5,3}.\vec{\nabla}\phi_{5,3}+\vec{\nabla}\phi_{5,4}.\vec{\nabla}\phi_{5,4} \end{cases} \tag{3.63}$$

The gradients of the basis functions are given by:

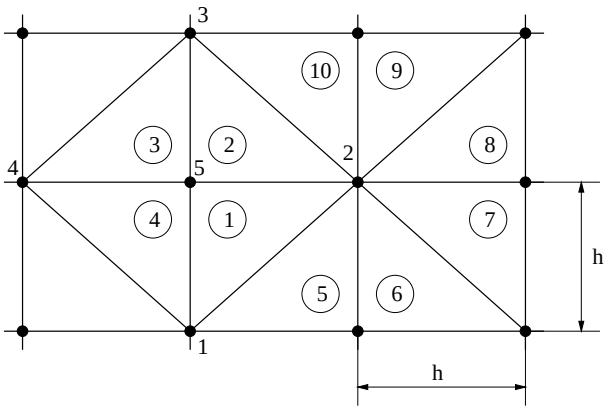


Figure 3.34: Example of a mesh leading to an inconsistent scheme

$$\begin{aligned}
&\vec{\nabla}\phi_{1,1} = |_{-1/h}^{0} && \vec{\nabla}\phi_{1,4} = |_{-1/h}^{0} \\
&\vec{\nabla}\phi_{2,1} = |_{0}^{1/h} && \vec{\nabla}\phi_{2,2} = |_{0}^{1/h} \\
&\vec{\nabla}\phi_{3,2} = |_{1/h}^{0} && \vec{\nabla}\phi_{3,3} = |_{1/h}^{0} \\
&\vec{\nabla}\phi_{4,3} = |_{0}^{-1/h} && \vec{\nabla}\phi_{4,4} = |_{0}^{-1/h} \\
&\vec{\nabla}\phi_{5,1} = |_{1/h}^{-1/h} \quad \vec{\nabla}\phi_{5,2} = |_{-1/h}^{-1/h} && \vec{\nabla}\phi_{5,3} = |_{-1/h}^{1/h} \quad \vec{\nabla}\phi_{5,4} = |_{1/h}^{1/h}
\end{aligned} \tag{3.64}$$

Then, we obtain the values of the coefficients a_i:

$$a_1 = a_2 = a_3 = a_4 = -\frac{2}{h^2} \quad \text{and} \quad a_5 = \frac{8}{h^2} \tag{3.65}$$

Let us calculate the contribution of triangle τ_2 to the right hand side of eq. (3.62) assuming that the source term f is constant over the whole domain Ω:

$$\int_{\tau_2} f\phi_{5,2} d\tau_2 = b_2 f \tag{3.66}$$

For this triangle, the basis function $\phi_{5,2}$ is written:

$$\phi_{5,2}(x,y) = \frac{1}{h}(h - x - y) \tag{3.67}$$

and the expression of the coefficient b_2 is given by:

$$b_2 = \frac{1}{h} \int_{x=0}^{x=h} \int_{y=0}^{y=h-x} (h - x - y)\, dxdy \tag{3.68}$$

that is:

$$b_2 = \frac{h^2}{6} = \frac{S}{3} \tag{3.69}$$

The calculation of the other terms of the right hand side of eq. (3.62) yields the same result. We finally obtain:

$$\sum_{i=1} 4 \int_{\tau_i} f \phi_{5,i} s\tau_i = \frac{4S}{3} f \tag{3.70}$$

From the previous result and taking into account a simplification by S and a division by 2, we obtain a discrete equation at node 5 which is very similar to the one resulting from a finite difference type method:

$$-\frac{1}{h^2}\left(u_1 + u_2 + u_3 + u_4 - 4u_5\right) = \frac{2}{3} f \tag{3.71}$$

Because the mesh depicted on figure 3.34 is a cartesian mesh, we can easily expand u_1, u_2, u_3 and u_4 in Taylor's series and replace the resulting expressions in eq. (3.71). By this mean, we can show that the local finite element discretization at node 5 is not consistent because of the coefficient $\frac{2}{3}$ in the right hand side of eq. (3.71). Moreover, if we write the finite element discretization at node 2, we obtain a right hand side equal to $\frac{4}{3} f$. This is because, in that case, node 2 belongs to 8 triangles while node 5 only belongs to 4 triangles. Fig. 3.35 shows a mesh configuration that leads to a consistent finite element discretization. Thus, the finite element formulation is not node-wise consistent depending on the underlying mesh. This means that if an exact continuous solution u is injected in a mesh τ, the residual $(R_k(u))_i$ on a given node may not tend to zero with h. This does not prevent the scheme from being convergent but makes difficult the use of the residual to monitor the FMG strategy.

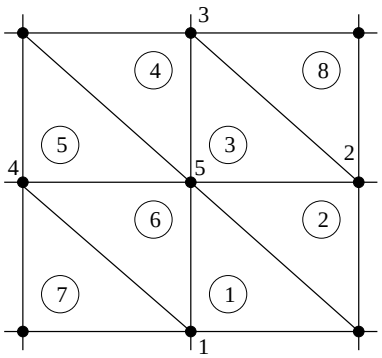


Figure 3.35: Example of a mesh leading to a consistent scheme

3.3.2.4.4 Residual monitored FMG strategy. In [24], Carré and Dervieux propose a functional treatment of this problem in the context of a multigrid by volume agglomeration method. The main idea is to remap the residual in a space allowing its estimation with a L^2 norm. This is done by introducing a *coarse space of smooth functions*:

$$W_H = \mathsf{span}[\chi_1 \ , \ \chi_2 \ , \ \cdots \ , \ \chi_N] \tag{3.72}$$

The symbol H indicates that we consider an approximation space for function defined on Ω, independently of the mesh size h. These functions are defined on a coarse mesh of characteristic size H such that $\forall k \ : \ h_k << H$. Assuming that W_H is generated by an orthonormal basis $(\chi_k)_k$ and that the functions χ_k are smooth, one can define a projection operator S_H from a distribution space D onto W_H by:

$$L_H w_k = \sum_{i=1}^{N} (w_k \ , \ \chi_i)\chi_i \tag{3.73}$$

Let us now consider the actual discrete residual at the beginning of a FMG phase for solving the discrete system $A_h u_h = f_h$ (resulting from the discretization of the Poisson problem (3.60)):

$$R_k = A_k P_{k+1}^k u_{k+1} - f_k = A_k (P_{k+1}^k u_{k+1} - u_k) \tag{3.74}$$

where u_{k+1} is the solution obtained on grid level $k+1$ and u_k is the fully iteratively converged solution of the discrete system on grid level k. In the present context, R_k is considered as belonging to the distribution space D.

Then, it is shown in [24] that the following estimation is valid:

$$\| L_H \left(A_k (P_{k+1}^k u_{k+1} - u_k) \right) \|_{L^2(\Omega)} \leq K h_k^p \tag{3.75}$$

In other words, the function $\bar{R}_k = L_H R_k$ is an approximation of the actual residual R_k which satisfies the estimate $\| \bar{R}_k \| \leq K h_k^p$. In [27], we describe how this result can be used to design a *residual monitored* FMG strategy, inspired from the method proposed by Carré and Dervieux[24]. In practice, the operator L_H is defined as:

$$L_H R_k = D_k^{k+1} D_{k+1}^{k+2} \cdots D_{K-1}^K$$

Let us assume that we are face with a second order accurate discretization scheme ($p = 2$) and that the coarsening ratio is equal to $\frac{h_{k+1}}{h_k} = 2$. From this point, $\| \bar{R}_k \|$ is called the *relative residual norm*. We now consider the case where the discrete system is fully converged at each FMG phase. In these conditions, one should observe that the relative residual norm is such that:

$$\frac{\bar{R}_k^{(0)}}{\bar{R}_{k+1}^{(0)}} < \frac{1}{4} \tag{3.76}$$

In the case of an incomplete convergence, this ratio has to be controlled. If the initial relative residual norm $\bar{R}_k^{(0)}$ is not 4 times smaller than that of the previous FMG phase (i.e. $\bar{R}_{k+1}^{(0)}$) then the convergence process has to be extended in order to compensate for the poor initial guess. This is performed by first requiring additional iterations until the relative residual norm reaches the expected value (i.e. divided by 4 with reference to the initial relative residual norm obtained at the previous FMG phase) and then continuing the iterations until the relative residual is further divided by 10. The main steps of the residual monitored FMG algorithm are the following:

1. Estimate the initial residual: $\bar{R}_k^{\text{init}} = \bar{R}_k^{(0)} = L_H R_k^{(0)}$
2. Estimate C_0 such that: $\| \bar{R}_k^{\text{init}} \|_{L^2} = \frac{C_1}{4} \| \bar{R}_{k+1}^{\text{init}} \|_{L^2}$ with $C_0 = \max(1\ ,\ C_1)$
3. Stop the the MG cycling when: $\| \bar{R}_k^{\text{final}} \|_{L^2} = \| L_H R_k^{\text{final}} \|_{L^2} \leq \frac{1}{C_0} \times \frac{1}{10} \| \bar{R}_k^{\text{init}} \|_{L^2}$

Note that, on one hand, one needs at least two grid levels k and $k+1$ to be able to evaluate the relative residual $\bar{R}_k$ and, on the other hand, the test in 3. can only be applied for the first time when starting the computation on the third grid level. Convergence to zero machine is not a costly operation on the two first coarse grid levels. Figure 3.36 illustrates the residual monitored FMG algorithm as exposed above.

3.3.3 Generation of the multigrid hierarchy

In contrast to structured grids, the use of unstructured meshes offers some major advantages in the mesh generation process. Moreover, it allows a larger flexibility in adapting the mesh to complex geometries and solutions. As a consequence, this type of data representation is now a common tool in CFD. However, this approach also requires sophisticated discretization methods and solution algorithms. As a matter of fact, classical discretization schemes and solvers using the regularity of the mesh may fail or become less efficient on unstructured meshes. This is in particular the case for multigrid methods that were originally formulated for structured grids. To run efficiently on non-structured meshes, the solution algorithms have to be adapted to the irregularity of the mesh. In structured multigrid algorithms, the building blocks of the methods are the inter-grid transfer and coarse grid operators. Unstructured multigrid algorithms add the additional difficulty of defining the coarse levels.

In the past ten years, two main multigrid techniques have appeared in CFD on non-structured meshes. The first one, already applied in structural mechanics, relies on the use of non-nested triangulations. The second technique is specific to CFD and is associated to finite volume discretization and agglomeration/aggregation

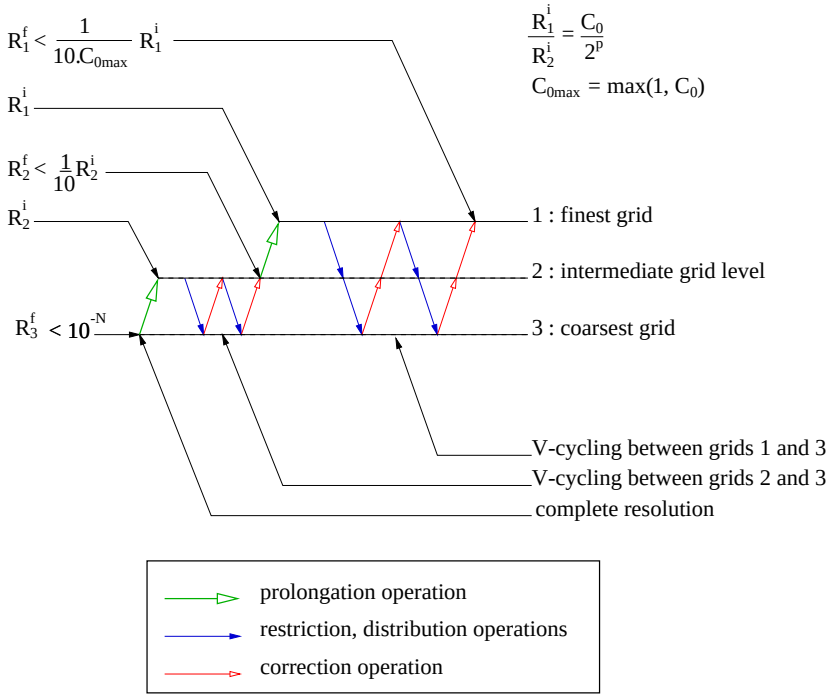


Figure 3.36: Principle of the residual monitored FMG algorithm

techniques. This approach has been considered in details in sections 3.1 and 3.2. For the first class of methods that explicitly define a hierarchy of grids, an important practical problem is to construct a sequence of grids with different resolutions. A coarse to fine path can be used, i.e., fine levels are generated by successive refinements starting from a given coarse mesh. While this approach possesses the advantages of simplicity, it requires that the coarse level is sufficiently large to adequately represent the geometry and that the successive refinements do not introduce artificial features in the geometry (for instance sharp edges). Moreover, in three dimensions, successive refinements of tetrahedra results in a loss of the quality of the mesh in such a way that the overall quality of the fine mesh can be rather poor. These reasons explain why an efficient multigrid strategy cannot be entirely based on successive refinements and that these methods have to be complemented by other techniques to generate the coarse meshes. An alternative is to generate all the mesh levels independently. However, this method places an excessive burden on the mesh generation step with the result that industrial users are not able to use multigrid in a production setting. Thus for practical purpose, one has to derive strategies for the automatic construction of coarse grids from a given non-structured one. In this section, we present several ways to perform this task. While the problem in two dimension appears to be relatively simple, this is not the case in a three dimensional setting and practical algorithms have appeared only very recently. We review some of them with a particular emphasis on the node nested strategy first advocated in [67].

3.3.3.1 Nested mesh approach

Let Ω be a bounded domain of $\mathbb{R}^d (d = 2, 3)$, and consider a coarse triangulation τ_1 of this domain defined by the set of nodes $\mathcal{N}_1$. Associated to this triangulation is the finite element space of piecewise linear functions $\mathcal{M}_1$. The spaces $\mathcal{M}_j$ will be recursively defined by adding nodes at the midpoints of the edges of the simplices of τ_{j-1} and decomposing each simplex into 2^d simplices. Hence, each element of $\mathcal{M}_j$ belongs to $\mathcal{M}_{j+1}$ and thus we obtain a sequence of nested spaces:

$$\mathcal{M}_1 \subset \mathcal{M}_2 \subset \cdots \subset \mathcal{M}_n \tag{3.77}$$

To connect the different levels, we need linear operators between them. The prolongation operator is simply the identity, and the restriction operator R^j_{j+1} can be defined by the L^2 projection on $\mathcal{M}_j$:

$$(R^j_{j+1}\ u_{j+1},\ v_j)_{L^2} = (u_{j+1},\ v_j)_{L^2} \tag{3.78}$$

The previous definitions are very simple and gives everything we need to define a multigrid method. The coarse grid operators can be defined by a variational formulation:

$$A^{j-1} = R^{j-1}_j A^j P^j_{j-1} \tag{3.79}$$

but, for CFD applications, are usually defined from a discretization of the coarse grid τ_{j-1}. However, for $d = 3$, this method has some major drawbacks that make it unsuitable for practical applications. First, it implies a large dependence of the fine mesh node distribution on the coarsest level. Indeed, the mesh division algorithm is a mesh generation algorithm, and unfortunately this is a very poor one. The point is that, in contrast to the 2D case where the refinement of a triangle, produces four congruent triangles, the subdivision of a tetrahedron produces four congruent tetrahedra but also four additional tetrahedra with deteriorated aspect ratio. As an illustration of this effect, figure 3.37 shows the deterioration of a sequence of tetrahedra generated by successive division from an equilateral tetrahedra. In table 3.17 is indicated the quality of the tetrahedra shown in figure 3.37. It is clear that the meshes generated this way are of poor quality and hence, the fine mesh solution will also be of poor quality.

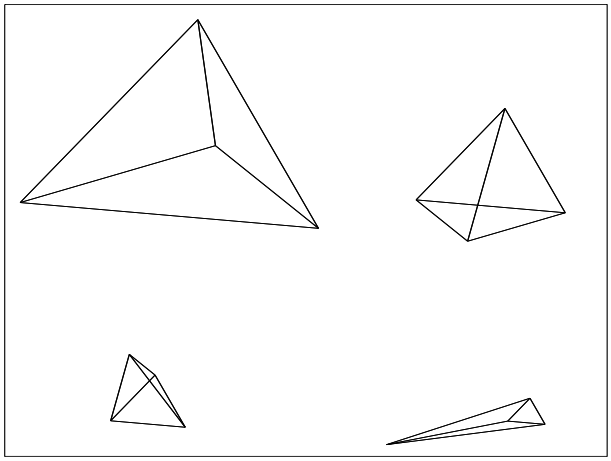


Figure 3.37: Deterioration of the mesh quality by element refinement

Table 3.17: Quality of Tetrahedra generated by successive refinement

	Fine level	First refined level	Second refined level	Third refined level
Mesh quality	1.0	0.9	0.79	0.66

A second drawback of the mesh refinement multigrid algorithm is that this technique requires that the coarse initial mesh is already fine enough to describe properly the geometry. Moreover if the geometry is described by a curved boundary, it is necessary to project the fine level boundary nodes on the boundary. If this is not done, the fine mesh geometry will contain artificial sharp edges resulting in a decrease of the solution quality or in the creation of geometrical artifacts in the solution. To illustrate this point, consider in figure 3.38 the Mach number distribution on an `ONERA M6` wing with a 0.84 inflow Mach number and a 3.06° angle of attack. One can observe large irregularities in the Mach number distribution on the surface of the wing. Comparison with the mesh (figure 3.38 right) shows that these irregularities are totally artificial and are due to the mesh refinement process. Therefore, to avoid this effect, the boundary curvature must be represented in accordance to the resolution of

the current mesh. This imposes that the exact geometry is stored either by an additional surface fine mesh or by a CAD system. Note also that the projection of the boundary nodes on the true surface can result in extremely badly shaped elements. This is illustrated in figure 3.39

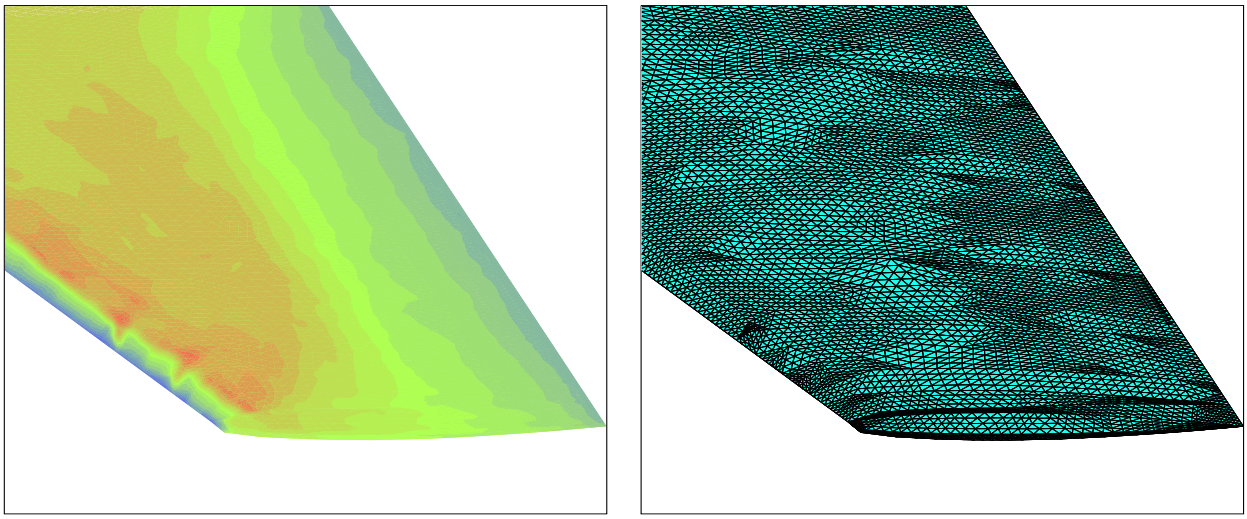


Figure 3.38: Mach number distribution and mesh on ONERA M6 wing

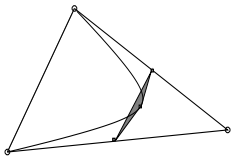


Figure 3.39: Creation of badly shaped element by node projection

3.3.3.2 Automatic coarsening of unstructured meshes

In the previous subsection, we have explained why a practical multigrid method cannot be solely based on successive refinements. Multigrid methods have to be complemented by other techniques to generate the coarse meshes. A solution is to relax the constraint on the way meshes are nested. The most radical method in this direction considers that the coarse and fine triangulations are generated independently using any given mesh generators. In CFD, this not-nested multigrid method appeared to be one of the most successful strategies and has been successfully used by several authors (see for instance the work of Mavriplis *et al.*[99]). In this approach the solution, residuals, and corrections are transferred back and forth through the different levels using linear interpolation between two successive levels. This introduces an additional complexity in the multigrid approach because the transfer operators between the different levels are now difficult to compute. Regardless of the accuracy of the transfer operators, one must determine the simplices where the nodes of the other levels are located. Efficient search algorithms exist for this purpose. However, this increases the cost and complexity of the multigrid algorithm. Moreover, the need to generate multiple meshes of the same geometry (that have in addition to respect some ratio between the discretisation parameters of the different levels) results in an excessive burden on the user. Clearly, a method that relies on the use of many independently generated meshes is hardly exploitable in an industrial environment. Consequently, algorithms that consider the generation of the coarse grid levels as part as the solution process have to be developed to make multigrid techniques practical.

Recently, different methods have been proposed to perform this task. Several algorithms use the concept of Maximal Independent Set[67]-[28]-[3] to generate a cloud of coarse grid nodes that are reconnected by a Delaunay

algorithm. Application of this method in CFD can be found in [104] for Euler computations and in [103] for Euler and Navier-Stokes ones. An alternative is to use edge collapsing algorithm [106]. However, all these methods experience difficulties in the respect of the boundary. The coarsening algorithm proposed here is based on a local mesh optimization primitive that includes the respect of the boundary in a very simple way. A 3D simplicial mesh is composed of vertices, edges, faces and tetrahedra. For each of these objects, it is possible to define a local operation consisting in removing this object from the mesh (considered as a graph) and re-meshing the shell formed by the deletion of this object. From the point of view of mesh coarsening, the only interesting primitive is the deletion of a node. Assume that to each node s_i of the fine triangulation is associated a scalar $h(i)$ that represents the desired length of the edges connected to s_i. Let $\mathcal{E}(i)$ be the set of nodes connected to node s_i and let $\mathcal{F}(i)$ be the set of exterior faces of the tetrahedra that contain node s_i. The re-meshing of the shell can be performed by exploring successively all the triangulations formed by connecting an arbitrary node in $\mathcal{E}(i)$ to the faces of $\mathcal{F}(i)$. From all these triangulations, we retain the ones whose volume is minimal (and equal to the original volume of the shell) and from the set of remaining triangulations, we retain the one that minimize a criterion based on the desired length $h(j)$ of the edges connected to the nodes $s_j \in \mathcal{E}(i)$. Note that in practice, we only explore a partial list of the possible tetrahedrizations of the shell, however experiments show that this partial exploration is sufficient to detect an acceptable tetrahedrization. The deletion of a boundary node is done with the same algorithm by using a trick due to Coupez and Chenot[34]. Let s_i be a boundary node, then the set of nodes $\mathcal{E}(i)$ connected to s_i is completed by a fictitious node whose coordinates are those of s_i. Tetrahedra (of null volume) are formed by connecting this fictitious node to the boundary faces containing s_i, embedding s_i into an artificial tetrahedrization. Then the node deletion algorithm is used as for true interior nodes. The only modification to the original algorithm stands in the fact that up to a prescribed tolerance, the volume (equal to zero) of the tetrahedra connected to the fictitious node is preserved in the new tetrahedrizations.

As an illustration, the proposed algorithm has been applied to the coarsening of tetrahedral mesh around a `Falcon` aircraft. The initial mesh consists of 30,514 nodes and 163,732 tetrahedra. We present in table 3.18 the results of the coarsening algorithm for different values of the *coarsening ratio* which is defined as the ratio between the desired length of the edges to the length of the edges of the initial tetrahedrization. Thus with a coarsening ratio of 2.0, the algorithm will try to generate a mesh where the edges connected to the remaining nodes have a length twice the original length. Table 3.18 shows that the coarsening algorithm is able to generate meshes with a large reduction of the number of nodes. Figure 3.40 shows that this is done with little influence on the geometry that appears to be preserved by the algorithm. For a large value of the coarsening ratio (> 2.5), the algorithm stalls and the number of points remains larger than what is requested.

Table 3.18: Meshes generated using different values of the coarsening ratio
N_V : # vertices - N_T : # tetrahedra

N_V	N_T	Coarsening ratio	Measured coarsening ratio
7728	37853	1.5	1.51
3813	16519	2.0	2.00
2913	15960	2.2	2.19
1982	8437	3.0	2.48
1982	8071	3.0	2.48

3.3.4 Localization problems

Since in this study the FAS and FMG multigrid algorithms are developed with no assumption regarding the link between tetrahedral meshes in the multigrid hierarchy, an appropriate (pre-processing) tool has been designed for solving the localization problems that yield the data required for implementing the inter-grid transfer operators. Here, we describe the main principles of this tool. Two localization problems must be solved:

L1 : find the identifier of the tetrahedra of the fine grid G_k that surrounds the vertex n_i^{k+1} of the coarse grid G_{k+1} i.e. $T_k(n_i^{k+1}) \in \tau_k$,

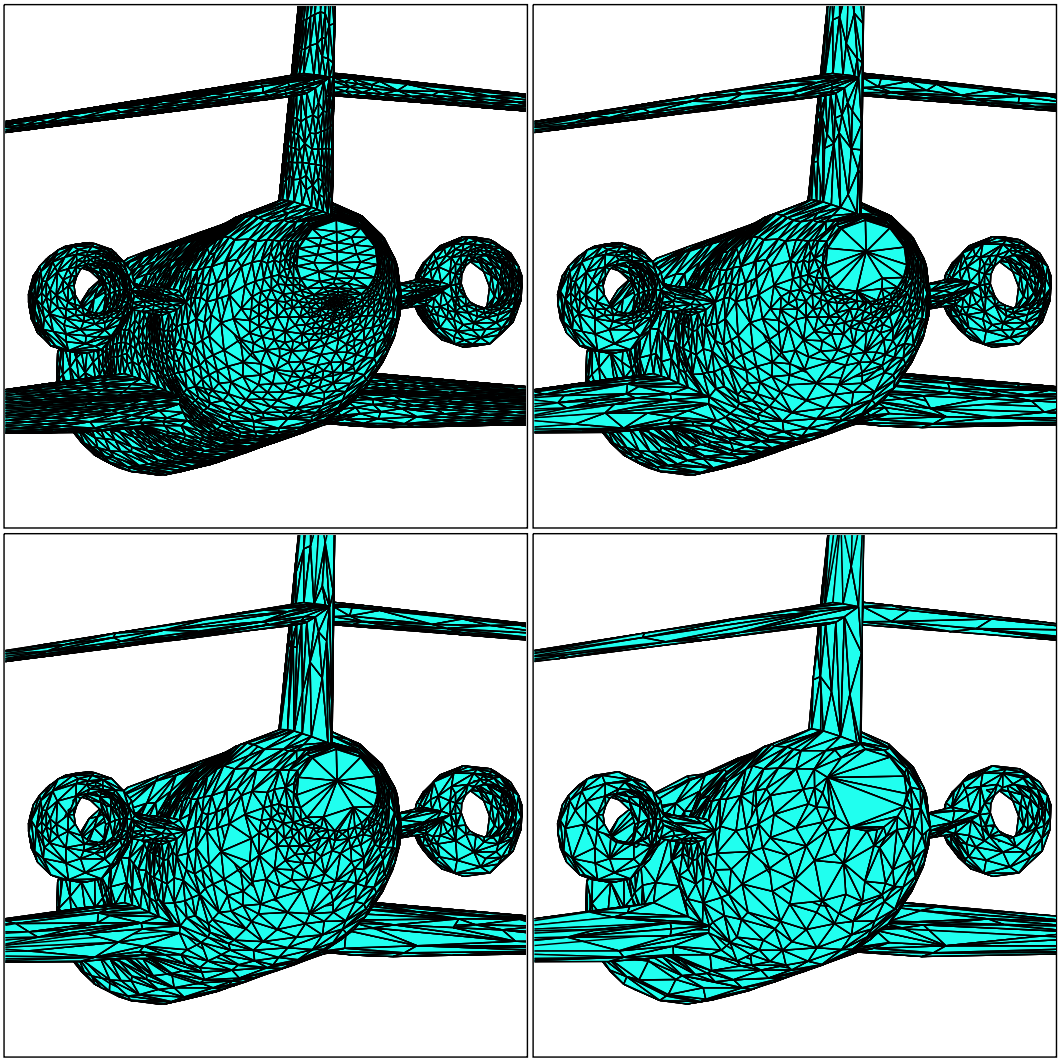


Figure 3.40: Rear view of the `Falcon` aircraft

L2 : find the identifier of the tetrahedra of the coarse grid G_{k+1} that surrounds the vertex n_i^k of the fine grid G_k i.e. $T_{k+1}(n_i^k) \in \tau_{k+1}$.

The resolution of these two localization problems can be very costly when the grids have a large number of vertices and elements. Therefore, specific strategies must be taken into account when scanning through the mesh elements while trying to locate a given point. Let us consider two grids: a fine grid FG and a coarse grid CG. Both localization problems can be solved using the same function with the two grids as parameters. However one particular problem often arises: a FG vertex can be outside the envelope of the coarse grid. Similarly, a CG node can be outside the envelope of the fine grid. This means that a localization algorithm based on the search of the element that strictly surrounds a given vertex can fail. For this reason, the resolution of a localization problem is decomposed in several steps that use different algorithms. Each algorithm has specific characteristics:

LOCALG1 : a search algorithm that requires $O(N_l)$ operations where N_l is the number of vertices to be localized.

LOCALG2 : a search algorithm that requires $O(N_r \times N_e)$ where N_r is the number of vertices that have not been localized by algorithm **LOCALG1** ($N_r << N_l$) and N_e the number of elements of the second mesh.

LOCALG3 : a search algorithm that requires $O(N_o \times N_b)$ where N_o is the number of vertices that have not been localized by algorithm **LOCALG2** and N_b the number of boundary elements of the second mesh.

Here and in the following, the term "second mesh" is used to reference the mesh whose elements are tested in order to localize a given vertex (i.e. the fine grid for localization problem **L1** and the coarse grid for **L2**).

Algorithm LOCALG1 : let P denotes the vertex to be localized. Figure 3.41 shows the principle of this algorithm. The search is started with the element # 1 on figure 3.41. First, the inward normals $\vec{n_a}$, $\vec{n_b}$ and $\vec{n_c}$ of this element are evaluated. Then, the scalar products $\vec{n_a}.\overrightarrow{\vec{B}P}$, $\vec{n_b}.\overrightarrow{\vec{C}P}$ and $\vec{n_c}.\overrightarrow{\vec{A}P}$ are calculated. If these threes scalar products are positive then vertex P is inside the element. If one scalar product is negative then P is not located in the current element and another element must be tested. The next element is the neighboring one by the side corresponding to the negative scalar product. On figure 3.41, the negative scalar product is $\vec{n_a}.\overrightarrow{\vec{B}P}$ and element # 2 is selected. This process is repeated until element # 10 that surrounds vertex P is reached. However, situations where no solution is found can occur. An example of such a situation is shown on figure 3.42. **LOCALG1** can fail if an element is tested several times and an infinite loop appears (closed path). A failure can also happen when the path reaches a boundary. To overcome the problems of multiple testings and infinite loop, a constraint is added that limits the total number of tested elements (in practice this limit can be set around 50). Nevertheless, it is clear that the success or failure of the search depends on the first element to be tested. In order to increase the rate of success and decrease the number of tested elements for a single search, a wavefront scanning strategy has been implemented. This strategy makes use of the connectivity graph of the underlying tetrahedral mesh. The idea is to select the next vertex to be localized from the set of vertices connected to vertex P by a mesh edge. The vertices are organized by layers as illustrated on figure 3.43 for a regular triangular mesh. This wavefront scanning strategy is complemented by a good choice of the next starting element which is key feature for the success of the algorithm. In practice, this element is chosen as the surrounding element of a previously localized vertex Q belonging to the set of neighboring vertices of N.

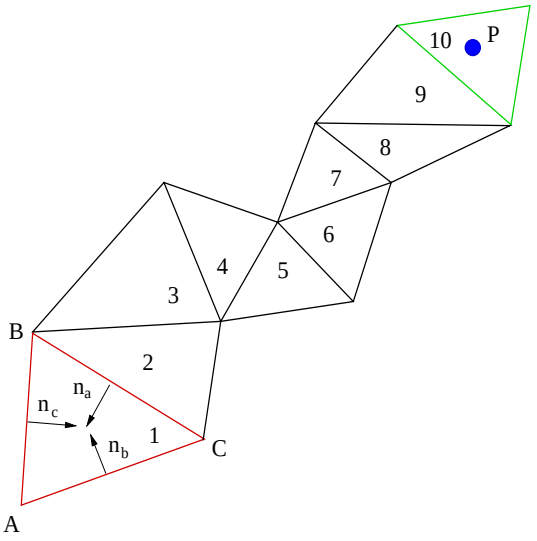


Figure 3.41: Principle of localization algorithm **LOCALG1**

LOCALG1 is an efficient localization algorithm and, in order to demonstrate that it requires $O(N_l)$ operations, it has been evaluated using a sequence of four not-nested tetrahedral meshes around an `ONERA M6` wing. The characteristics of these meshes are summarized in table 3.27. The localization tool is applied three times: the first run treats meshes MU3 and MU4, the second run treats meshes MU2 and MU3 while meshes MU1 and MU2 are treated in the third run. The results of the first localization problem **L1** are reported in table 3.19. In this

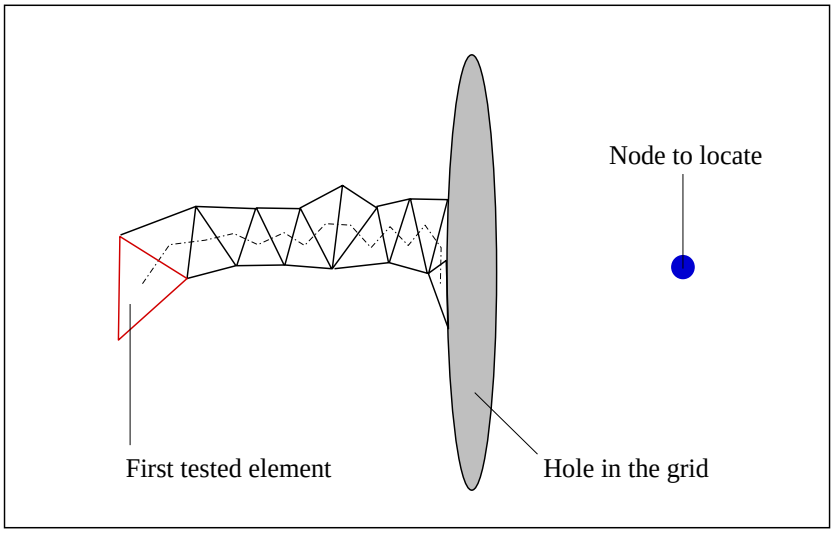


Figure 3.42: Failure situations with algorithm **LOCALG1**

table, "# loc. vert." denotes the number of vertices that are localized within a given step while "# tests" is the average number of tested elements. The first step ("Step 1" in the table), corresponds to the application of algorithm **LOCALG1**. The two other steps are performed using complementary algorithms to treat the vertices that have not been localized by algorithm **LOCALG1** (see the next paragraphs). These two steps are certainly not optimum but the reason why they are needed will be explained later. We note that, on average, only 7 or 8 fine grid elements are tested in the application of algorithm **LOCALG1**. This figure does not depend on the total number of fine grid elements and clearly shows that algorithm **LOCALG1** requires $O(N_l)$ operations. As can be seen in table 3.19, the CPU time requested to solve a localization problem using this algorithm is proportional to the number of vertices to be localized. For each pair of meshes, the rate of success of algorithm **LOCALG1** is respectively equal to 97.8%, 98.7% and 96.3%. Table 3.20 summarizes the results of the second localization problem. Similar conclusions can be drawn. In that case, For each pair of meshes, the rate of success of algorithm **LOCALG1** is respectively equal to 98.9%, 98.0% and 98.0%. The main difference with the treatment of the first localization problem is the average length of the path during the search. In the case of the first localization problem, the average length is equal to 7 or 8 elements whereas it is only equal to 3 or 4 elements for the second localization problem.

The fact that the path is shorter for the second localization problem is easy to explain and illustrated on figures 3.44 and 3.45. For the fine grid vertex a (see figure 3.44), the localization algorithm starts with the coarse grid element A that surrounds the already localized vertex c which is a neighbor of vertex a. Because the fine grid vertices a and c are close to each other and because the coarse grid element A is large in size compared to a fine grid element, A has a high probability to surround vertex a. Indeed, element A is the solution of this localization problem for vertex a. In the case of vertex b, the starting element A that surrounds its neighbor c is not the solution of the problem. But element C that surrounds vertex b is close to A and the path to go from A to C is very short. The average length of the path for the first localization problem can be explained in the same way (see figure 3.45). To localize the coarse grid vertex A, the algorithm starts with the fine grid element a that surrounds the already localized vertex B which is a neighbor of A. The element a is small in size compared to the size of a coarse grid element and it has a low probability to surround vertex A. Since A and B are distant compared to the size of element a, the path to go from a to f (the element that surrounds A) is obviously longer than in the case of the second localization problem.

In summary, the above results show that the cost of the resolution of the two localization problems using algorithm **LOCALG1** is linearly proportional to the number of vertices to be localized and essentially independent of the number of elements of the second mesh. The situations for which **LOCALG1** fails to localize a given

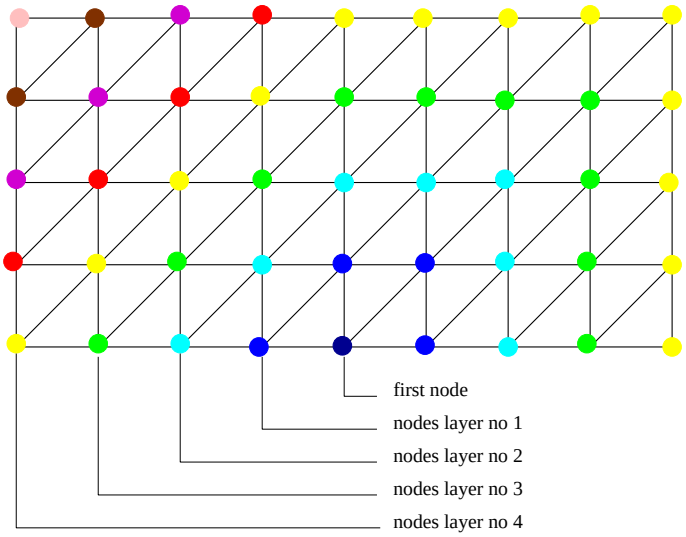


Figure 3.43: Wavefront scanning strategy in algorithm **LOCALG1**

Table 3.19: First localization problem **L1** using not-nested meshes around an `ONERA M6` wing

	Meshes MU3-MU4			Meshes MU2-MU3			Meshes MU1-MU2		
	# loc. vert.	# tsts	CPU	# loc. vert.	# tsts	CPU	# loc. vert.	# tsts	CPU
Step 1	92,458	8	4.0 sec	22,710	7	0.9 sec	5,185	8	0.3 sec
Step 2	2,034	-	1850.0 sec	297	-	73.0 sec	196	-	6.4 sec
Step 3	1,991	-	596.0 sec	270	-	42.0 sec	103	-	4.0 sec

vertex are the following:

1. the search path reaches a boundary,

2. the number of tested elements is larger than the allowable maximum value,

3. the vertex is outside of the envelope of the second mesh.

Situations 1. and 2. can be cured by running a new search with a better candidate for the starting element. However, our experiences have shown that such a strategy does not yield a satisfactory rate of success. Situation 3. cannnot be overcome using algorithm **LOCALG1**. Indeed, there is no way a posteriori to know if the search has failed due to a bad path or because the vertex is outside of the envelope of the second mesh. This is the reason why a different algorithm must be used to localize the remaining vertices.

Table 3.20: Second localization problem **L2** using not-nested meshes around an `ONERA M6` wing

	Meshes MU3-MU4			Meshes MU2-MU3			Meshes MU1-MU2		
	# loc. vert.	# tsts	CPU	# loc. vert.	# tsts	CPU	# loc. vert.	# tsts	CPU
Step 1	312,799	4	7.0 sec	92,560	3	2.0 sec	22,632	3	0.5 sec
Step 2	3,475	-	766.0 sec	1,932	-	77.0 sec	375	-	2.6 sec
Step 3	3,355	-	499.0 sec	1,597	-	60.0 sec	245	-	3.3 sec

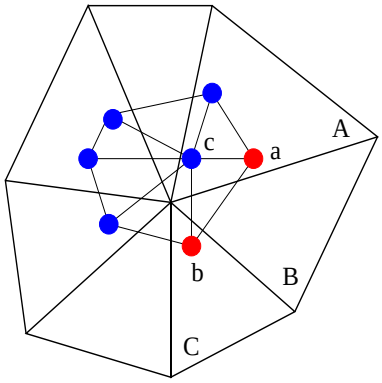


Figure 3.44: Illustration of a short **LOCALG1** path in the case of the second localization problem. The blue vertices are already localized. The starting element for the red vertex a is the element A that surrounds its neighbor c. Because this element is also the one surrounding vertex a, the **LOCALG1** path contains only element A. The starting element for the red vertex b is also element A. The **LOCALG1** path is constructed with the three elements A, B and C.

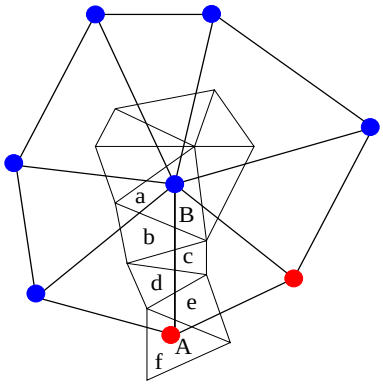


Figure 3.45: Illustration of a long **LOCALG1** path in the case of the first localization problem. The blue vertices are already localized. The starting element for the red vertex A is the element a that surrounds its neighbor B. The **LOCALG1** path is constructed with the six elements a, b, c, d, e and f, and is longer than the one obtained in the case of the second localization problem.

Algorithm LOCALG2 : once most of the vertices have been localized using algorithm **LOCALG1**, the remaining vertices can be split into two classes:

Class 1 : vertices not localized because of a failing **LOCALG1** search due to a bad path,

Class 2 : vertices that are outside of the envelope of the second mesh.

The aim here is to localize the remaining vertices that belong to Class 1. Because of the high rate of success of algorithm **LOCALG1**, the vertices that have not been localized by this algorithm are not numerous. Therefore, a costly but robust algorithm can be used while trying to minimize the increase in total computing time. This algorithm simply consists in testing, for each remaining vertex, all the elements of the second mesh until the one surrounding the vertex is found. If N_r is the number of vertices that have not been localized by algorithm **LOCALG1** and N_e the number of elements of the second mesh, such an algorithm requires $O(N_r \times N_e)$ tests. Timings for the meshes corresponding to table 3.27 are given in the "Step 2" entry of tables 3.19 and 3.20. Once this step has been performed, the remaining vertices belong to Class 2.

Algorithm LOCALG3 : the third step of the resolution of the localization problems aims at treating the vertices that are outside of the envelope of the second mesh. A specific method has been designed for this purpose. The objective is to decide to which element of the second mesh each vertex in Class 2 must be associated. The strategy adopted here associates such a vertex with the element of the second mesh that minimizes the deviation of the linear interpolation coefficients from the range [0.0 ; 1.0]. Because the vertices in Class 2 are outside of the envelope of the second mesh, the underlying search can be limited to boundary elements. If N_o denotes the number of vertices to be localized by algorithm **LOCALG3** and N_b the number of elements of the second mesh then this step requires $O(N_o \times N_b)$ tests. Timings for the meshes corresponding to table 3.27 are given in the "Step 3" entry of tables 3.19 and 3.20.

3.3.5 Parallel computing aspects

3.3.5.1 Parallelization of the monogrid solver

In the `THOR` code, parallel computing aspects are taken into account through the `Aztec`[71] library. `Aztec` contains a set of functions for defining and manipulating distributed sparse matrices together with implementations of iterative solvers and associated preconditioning techniques. The sparse matrix-vector product, $y \longleftarrow Ax$ is the major kernel operation of `Aztec`. The parallelization of `THOR` relies on the `Aztec` communication data structures and the high-level boundary exchange functions included in the `Aztec` library. To perform this operation in parallel, the vectors x and y as well as the matrix A must be distributed across the processors. For sparse irregular matrices, this is generally achieved using a graph partitioning tool such as `MeTiS`[80].

The main convention adopted in `Aztec` is that when calculating elements of y, a processor computes only those elements in y to which it has been assigned. These vector elements are explicitly stored on the processor and are defined by a set of indices referred to as the processor's $update$ set. The $update$ set is further divided into two subsets: $internal$ and $border$. A component corresponding to an index in the $internal$ set is updated using only information on the current processor. As an example, the index i is in the $internal$ set if, in the matrix-vector product kernel, the element y_i is updated by this processor and if each j defining a nonzero A_{ij} in row i is also in $update$. The $border$ set defines elements which would require values from other processors in order to be updated during the matrix-vector product. For example, the index i is in the $border$ set if, in the matrix-vector product kernel, the element y_i is updated by this processor and if there exists at least one j associated with a nonzero A_{ij} in row i that is not in $update$. In the matrix-vector product, the set of indices which identify the off-processor elements in x that are needed to update components corresponding to $border$ indices is referred to as $external$. These elements are stored locally and explicitly obtained from other processors (the $neighbors$) via communication whenever a matrix-vector product is performed. Figure 3.46 illustrates how a set of vertices in a partitioning of a grid would be used to define the $internal$, $external$ and $border$ sets. Clearly, this distribution of local data results in the use of one layer overlapped grid partitions.

In `Aztec`, the three types of vector elements are distinguished by locally storing the $internal$ components first, followed by the $border$ components and finally by the $external$ components. In addition, all $external$ components received from the same processor are stored consecutively. If the vector possesses a total of N elements, we then have the following repartition:

$[0, N_{internal} - 1\ ]$: $internal$ elements updated without communication;

$[N_{internal}, N_{internal} + N_{border} - 1\ ]$: $border$ elements updated with communication;

$[N_{internal} + N_{border}, N - 1\ ]$: not updated, but used to update $border$.

Similar to vectors, a subset of matrix non-zeros is stored on each processor. In particular, each processor stores only the rows that correspond to its $update$ set. For example, if vector element i is updated on processor p, then processor p also stores all the non-zeros of row i in the matrix. Further, the local numbering of vector elements on a specific processor induces a local numbering of matrix rows and columns: if vector element k is locally numbered as k_l, then all references to row k or column k in the matrix would be locally numbered as k_l. Thus each processor contains a sub-matrix which row and column entries correspond to variables defined on this processor.

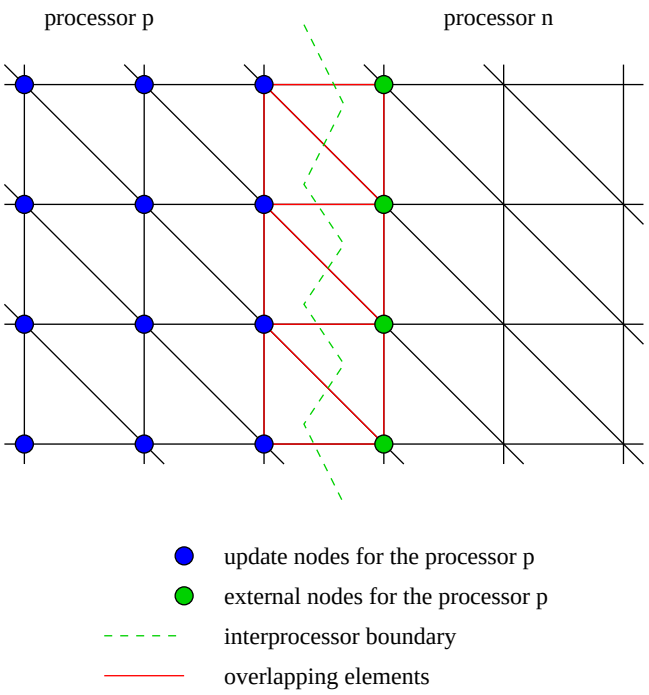


Figure 3.46: Example partitioning of a finite element grid in 2D

3.3.5.2 Parallelization of the multigrid solver

The parallelization of the multigrid algorithms developed in this study induces two types of communication operations:

intra-grid communications consist of the exchange of values at the interface between pairs of subdomain (these communications are generally referred as local communication operations). Clearly, on the coarse grid levels these communications can be performed by following the strategy adopted in the monogrid solver. In particular, the parallelization of the coarse grid calculations requires the partitioning of the corresponding meshes. This partitioning must be done with some level of consistency between the various coarse grids (i.e. the latter canot be partitioned independently) because particular criteria must be verified by the overall set of partitioned grids.

inter-grid communications are closely related to inter-grid transfer operations. They occur between two consecutive grids. These communication steps can be identified (i.e. the corresponding data structures can be constructed) using the partitioning of the involved grids on one hand, and, on the other hand, the result of the localization problems. Recall that the use of not-nested grids in the multigrid hierarchy requires to solve these localization problems for the construction of inter-grid transfer operators.

In IDeMAS, a pre-processing tool has been developed in order to construct the partitions of the coarse grids of a multigrid hierarchy given the partition of the finest grid. The goals are to minimize the cost of the communications for the parallel version of the transfer operators (inter-grid communications) as well as the cost of the communications on a given grid level (intra-grid communications), and to optimize the computational load balance between the processors.

The inter-grid transfer operators are built on a geometric basis. This means that a transfer between two grid levels will involve nodes and elements geometrically close to each other. Starting from this fact and keeping in mind that the goal is to minimize the interprocessor communications for the parallel version of the transfer operators, the proposed partitioning strategy is based on a geometric criterion. Let us consider a sequence of

possibly not-nested grids τ_k. Let τ_0 be the finest grid. The algorithm for the partitioning of the sequence of grids is the following:

1. Build the partition P_0 of the finest grid τ_0 with the MeTiS graph partitioner [80].
2. $k = 0$.
3. $k = k + 1$.
4. Solve the localization problem $L_{k,k-1}$ between τ_k and τ_{k-1}.
5. Build the partition P_k of the grid τ_k using $L_{k,k-1}$ and P_{k-1}.
6. If τ_k is not the last grid of the sequence go back to point 3).

Step 5) is performed using a geometric method in order to decide to which partition a coarse node will belong to. Recall that in the monogrid solver a node-wise partitioning is adopted. A node belongs to one partition and only one. Because of this choice, it is possible to find some elements that have nodes belonging to different partitions. These elements are called $overlapping\ elements$.

As pointed previously, a transfer between two grids involves elements and nodes close to each other. Typically, the basic operation of the solution restriction operator involves a coarse grid node and the fine grid element that surrounds this node. In order to minimize the cost of interprocessor communications for the parallel version of the transfer operators, one must try to have the coarse node and its surrounding fine grid element located in the same partition. This is the reason why the partitioning of a coarse grid is built with the help of the partitioning of the fine grid and with the help of the solution of the localization problem for these two grids. Indeed, the two localization problems needed to build the inter-grid transfer operators for the MG acceleration are the following:

1. find the fine grid element surrounding a coarse grid node;
2. find the set of fine grid nodes included in a coarse grid element.

For the partitioning of the coarse grid, we make use of the solution of the first localization problem. The algorithm is the following :

1. Loop over the coarse grid nodes: N_i.
   (a) Get the fine grid element t_j surrounding N_i.
   (b) Get the connectivity of the element t_j: n_0, n_1, n_2, n_3.
   (c) If the fine grid nodes n_j are all in the same partition then N_i is put in this partition.
   (d) If the fine grid nodes n_j are in different partitions then the coarse grid node N_i is given the partition number of the closest fine grid node n_j.

This algorithm is illustrated on figure 3.47. The left part of the figure shows the case where the fine grid element surrounding the coarse grid node is $an\ overlapping\ element$. As the graph of the coarse grid is not used, this partitioning method can generate a partition with isolated nodes. A node is said to be isolated if none of its neighbors is in the same partition. To correct this problem, a simple method consists in changing the partition number of the isolated node to the partition of one of its neighbors.

Because of the purely geometric criterion, the computational load associated to a coarse grid can be unbalanced. In order to obtain a well balanced computational load, a simple algorithm has been implemented in the pre-processing tool. Its principle is to move some nodes belonging to overlapping elements from an overloaded partition to an underloaded one. A partition is said to be overloaded when its number of nodes exceeds the ideal number of nodes calculated to have perfectly balanced processor loads. A node can be moved from a partition to another one only if it belongs to an overlapping element and if its partition number is different from the one of the three other nodes. The process is repeated until the correct computational load balance is achieved. Figure 3.48 illustrates this algorithm. Again, this algorithm can produce isolated nodes. To correct this problem, we have to change the partition number of these isolated nodes. The processor's load is then modified but tests performed with this method have shown very few changes.

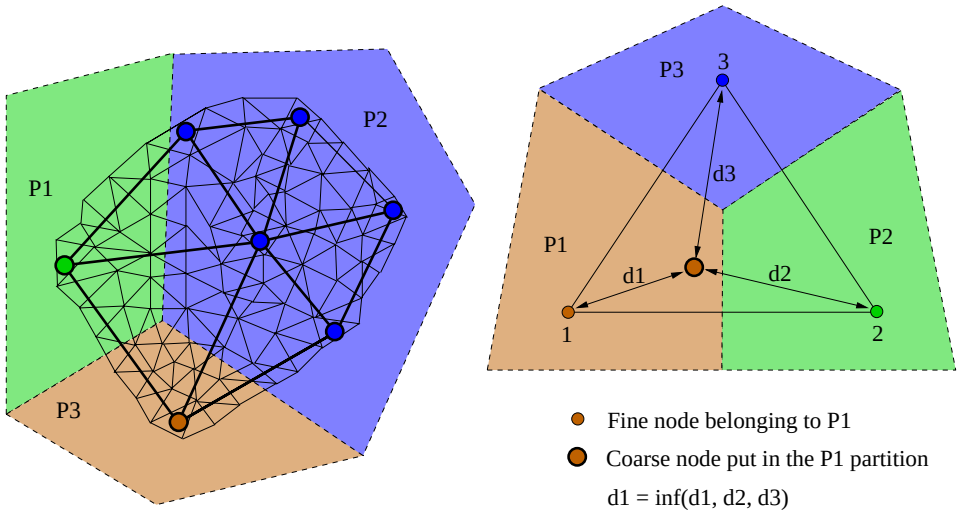


Figure 3.47: Partitioning of a coarse grid

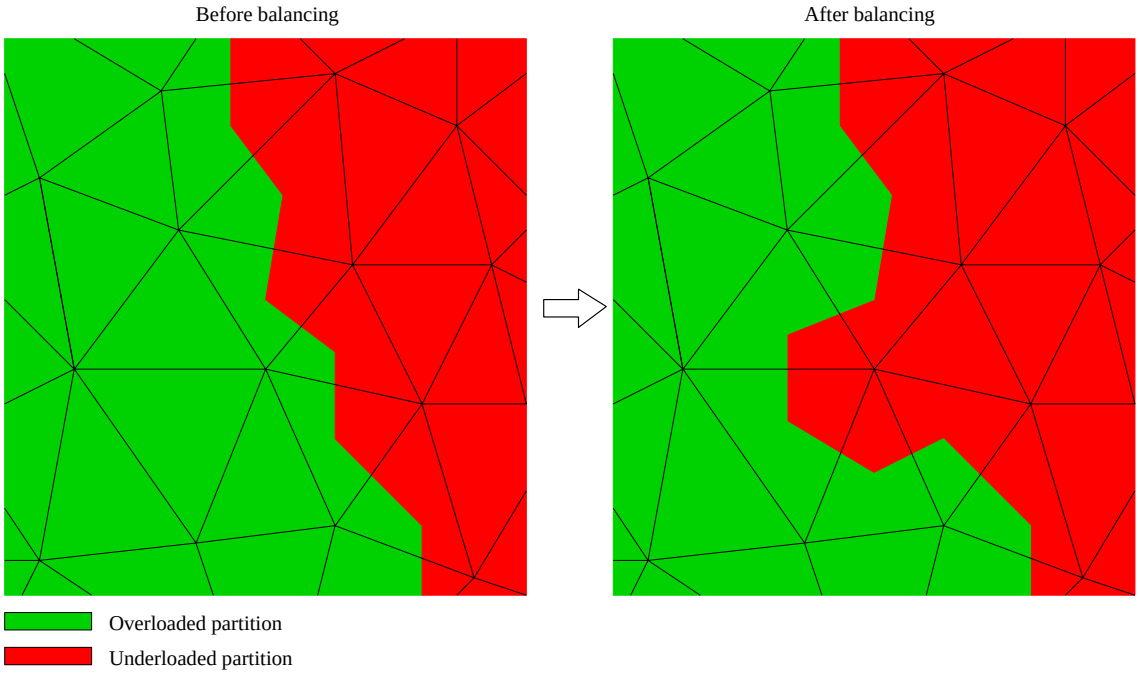


Figure 3.48: Load balancing of a coarse grid

So far, we have considered strategies for building coarse grid partitions such that, on one hand, the extra communication operations incurred by the parallel implementation of the inter-grid transfer operators are minimized and, on the other hand, the associated computational load is evenly balanced between the processors that are concerned by the result of the partitioning. Because it is not always possible to have a coarse grid node surrounded by a fine grid element of the same partition (see figure 3.49) and because a coarse grid element can include fine grid nodes from different partitions, interprocessor communications have to be performed for the parallel version of the inter-grid transfer operators. These operators are classified into two categories depending on the type of informations required for their parallel implementation:

1. the solution restriction operator;
2. the residual distribution, the solution correction and the solution prolongation operators. These operators belong to what we call *the prolongation operators class*.

In the following paragraphs, we detail the parallel implementation of the inter-grid transfer operators.

3.3.5.2.1 The parallel restriction operator. When the partitions are modified in order to insure a correct computational load balance, extra communications for the parallel version of the transfer operators are needed because some coarse grid nodes will be surrounded by fine grid elements of a different partition. Figure 3.49 shows such a situation. Originally, the coarse grid node n_1 belonged to the partition P_1 according to the algorithm used to build the coarse grid partition. But, because the partition P_2 was under-loaded and the partition P_1 was overloaded, node n_1 has been moved from P_1 to P_2. The same situation occurred for node n_3 moved form the partition P_3 to P_2. These two nodes now belong to the partition P_2 but are surrounded by fine grid elements that belong to different partitions.

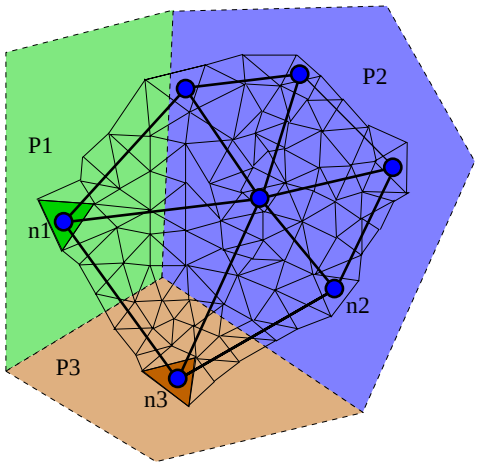


Figure 3.49: Consequence of the load balancing on the interprocessor communications for the solution restriction operator

The solution restriction operator uses the solution of the first localization problem i.e. a coarse grid node is associated with the fine grid element that surrounds it. As a general rule adopted in this study, the description of any transfer operator is done with reference to the coarsest of the two grid levels involved in the operation. This means that in the case of the restriction operator, the main loop is over the coarse grid nodes. For each coarse grid node, the index of the surrounding fine grid element is retrieved and the calculations are performed from the point of view of the coarse grid. To keep this unchanged for the parallel version of this operator, additional informations have to be stored. In particular, if a coarse grid node is surrounded by a fine grid element from a different partition as it is the case for the nodes n_1 and n_2 of figure 3.49, the global index of this element and its global connectivity are stored in an appropriate fine grid structure. Indeed, this element is added to the list of elements of the fine grid partition and the nodes of this element are added to the list of nodes.

3.3.5.2.2 The parallel distribution and prolongation operators. The solution prolongation, residual distribution and solution correction operators all require the same additional informations for their parallel versions. Figure 3.50 shows the situation which is faced in the case of the prolongation operator class. A coarse grid element contains fine grid nodes from different partitions. All the overlapping elements are concerned by this situation. More generally, any coarse grid element may be concerned by such a case due to the method to balance the processor load.

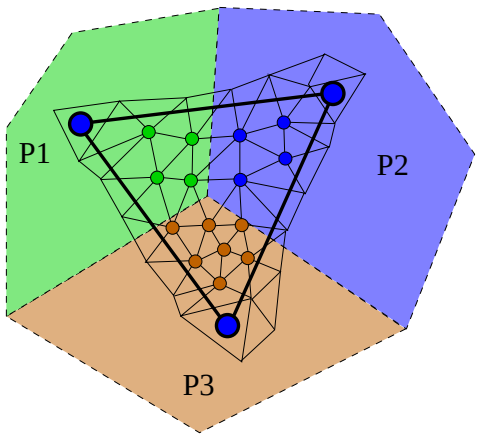


Figure 3.50: Consequence of the element overlapping between the partitions and of the load balancing on the interprocessor communications for the distribution operator class: a coarse grid element can surround fine grid nodes from different partitions

During the residual distribution operation, the solution of the second localization problem (fine grid nodes in a coarse grid element) is used in the following way: a loop over the coarse grid elements is done. For each coarse grid element, the number and the list of the fine grid nodes included in this element are retrieved. For each fine grid node in this list, its contribution to the total residual of each coarse grid node of the current coarse grid element is calculated by a summation. Once these two loops are achieved, the final value of the distributed residual is known for each coarse grid node. To keep the use of the solution of the second localization problem unchanged in order to make parallel multigrid solver easier to develop and to maintain, the fine grid nodes included in the coarse grid element of the partition P_2 (see figure 3.50) but that do not belong to this partition (the ones belonging to the partitions P_1 and P_3) are added to the list of nodes of the fine grid partition P_2.

The left part of figure 3.51 shows that some fine grid nodes from the partitions P_1 and P_3 contribute to the residual distribution at the coarse grid node n_2. The processor P_2 must have received the values of the residual at these fine grid nodes before starting the calculations. In this case, the required interprocessor communications have to be performed at the beginning of the residual distribution function before the main loop over the coarse grid nodes. The right part of figure 3.51 shows that the solution prolongation at the fine grid node f belonging to the partition P_1 is performed by the processor P_2 without the need to know anything except its coordinates before the calculation. In this case, the required interprocessor communications have to be performed at the end of the solution prolongation function. Figure 3.51 clearly shows that, depending on the transfer operator under consideration, the value at the fine grid node f from the partition P_1 has to be received by the processor P_2 (distribution case) or has to be sent by P_2 to P_1 (prolongation case).

The case of the solution correction operation is a little more complex as communications have to be performed before and after the correction. This is due to the fact that the solution at node f is corrected by a summation: $u_f = u_f + \Delta u_f$. Thus, the value of the solution at node f must be known by the processor P_2 before applying the correction. Once the solution at node f is corrected, it has to be send to the processor P_1 who owns this node.

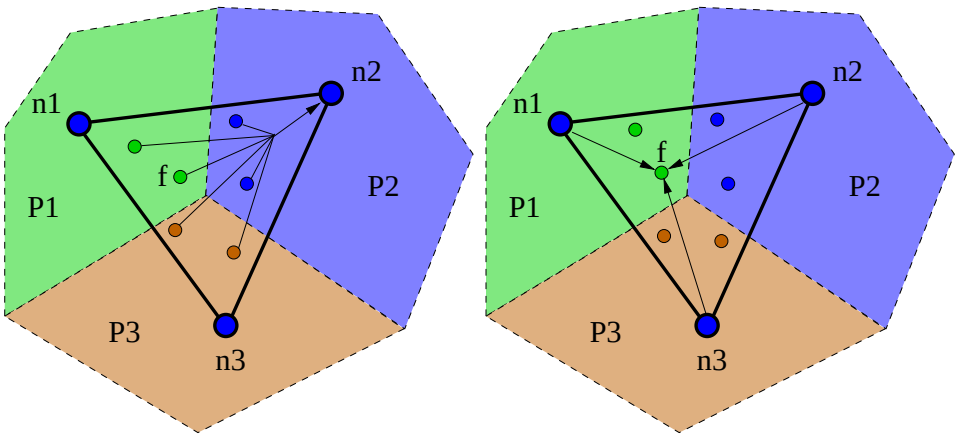


Figure 3.51: Contribution of the fine grid nodes from different partitions to the residual distribution to the vertices of a coarse grid element (left figure) and contribution of the vertices of a coarse grid element to the solution prolongation to fine grid nodes from different partitions (right figure)

3.3.6 Numerical and performance results

3.3.6.1 Computing platforms and conventions

Calculations have been performed on a SGI Origin 2000 MIMD system equipped with Mips R10000/195 Mhz processors. The code is written in Fortran 77 and the Mips F77 compiler has been used with maximal optimization options. The native SGI implementation of MPI has been used. Performance results are given for 64 bit arithmetic computations. In the following tables, N_p gives the number of processors for the parallel execution, N_g is the total number of levels in the multigrid hierarchy (finest mesh included). "Wall clock time" denotes the elapsed time of a simulation. The parallel speedup is denoted by $S(N_p)$. Finally, the term "linear threshold" is used to specify the accuracy of the linear system solves (i.e. the level of reduction of the initial linear residual) while the term "non-linear threshold" is used to characterize the convergence to the steady flow solution.

For all the numerical experiments considered below, the simulation is started from a uniform flow. These simulations aim at assessing the numerical and parallel efficiencies of the non-linear multi-mesh FAS-MG and FMG algorithms for fully nested as well as not-nested multigrid hierarchies.

3.3.6.2 External flow around an `ONERA M6` wing: fully nested meshes

The FAS-MG and FMG algorithms are first assessed by considering the classical test case given by the inviscid transonic flow around an `ONERA M6` wing. The free-stream Mach number is set to 0.84 and the angle of attack to $3.06°$. The numerical experiments considered here involve a sequence of three meshes whose characteristics are given in table 3.21. Mesh M2 (resp. M3) has been obtained through a uniform division of mesh M1 (resp. M2). As a consequence these meshes are fully nested i.e. node and element nested.

Table 3.21: Characteristics of fully nested meshes around an `ONERA M6` wing
N_V : # vertices - N_T : # tetrahedra - N_F : # boundary faces

Mesh	N_V	N_T	N_F
M1	2,203	10,053	2,004
M2	15,460	80,424	8,016
M3	115,351	643,392	32,064

We report on three series of calculations that have been performed for the following approximation schemes: the first order N-scheme (see paragraph 1.2.2.2.1 in section 1.2.2 of chapter 1), the second order PSI-scheme (see paragraph 1.2.2.2.2 in section 1.2.2 of chapter 1) and the second order PSI-scheme applied to the preconditioned Euler equations (see subsection 1.2.2.4 in section 1.2.2 of chapter 1).

3.3.6.2.1 Calculations based on the first order N-scheme. Several solution strategies are compared:

- *monogrid calculation.* A reference monogrid calculation has been performed on the finest mesh. The semi-discrete equations are time integrated using a linearized backward Euler implicit scheme (the CFL has been fixed to 100). At each time step, the resulting linear system is approximately solved using the GMRES method combined to a BMILU preconditioner (the linear threshold has been set to $\varepsilon_l = 10^{-1}$). The BMILU (Block Modified ILUP method is a version of the standard BILU(0) (Block ILU(0)) that only requires the storage of an additional block diagonal matrix for the preconditioner instead of a matrix of the size of the original matrix, which can be too memory intensive for large 3D problems. For what concern the convergence to steady-state, the non-linear threshold has been fixed to $\varepsilon_{nl} = 10^{-9}$.

- *FAS-MG calculations.* The multigrid hierarchy is composed of the three meshes whose characteristics are given in table 3.21. We compare the numerical and parallel efficiencies of two FAS-MG strategies respectively based on a V-cycle and a F-cycle, each of them involving 2 pre- and post-smoothing steps on the fine and intermediate grid levels and 4 smoothing steps on the coarsest grid level. In the present case, one application of the smoother consists in one time step of the backward Euler implicit scheme adopted for the monogrid calculation.

- *Controlled FMG calculations.* As previously, we compare the performances of two FMG strategies respectively based on the V-cycle and F-cycle defined previously, all parameters being unchanged except for the CFL which has been set to 50. In the initial phase of these FMG strategies, the problem defined on the coarsest mesh is solved using the backward Euler implicit scheme with a non-linear threshold fixed to $\varepsilon_{nl} = 10^{-9}$.

On figure 3.52 we compare the convergence profiles associated to the monogrid and FAS-MG strategies while figure 3.53 is dedicated to the FMG strategies. Performance results are given in table 3.22. In this table, the fourth column must be interpreted as follows :

- for the monogrid algorithm (SG), this column gives the total number of time steps performed for the convergence to the steady flow solution (i.e. to reach the required non-linear threshold);

- for the FAS-MG multigrid algorithms (MG-V and MG-F), this columns gives the total number of cycles to obtain the steady flow solution;

- for the FMG algorithms (FMG-V and FMG-F) the corresponding entries are of the form $n_G/n_2/n_3$ where n_G is the number of backward Euler implicit time steps that have been performed on the coarsest grid level while n_2 and n_3 respectively denote the number of two-grid and three-grid FAS-MG cycles.

For this first order calculation the monogrid algorithm appears to be an efficient solution strategy and the MG-V and MG-F multigrid algorithms do not result in computing time savings despite a notable difference in the convergence behaviors. The FMG-V strategy is 4.7 times faster than the monogrid algorithm on 8 processors. Finally, for both the FAS-MG and FMG algorithms, the V-cycle is always outperforming the F-cycle in terms of computing times.

3.3.6.2.2 Calculations based on the second order PSI-scheme. As previously, several solution strategies are compared:

- *monogrid calculation.* A reference monogrid calculation has been performed on the finest mesh. The semi-discrete equations are time integrated using a linearized backward Euler implicit scheme (the CFL

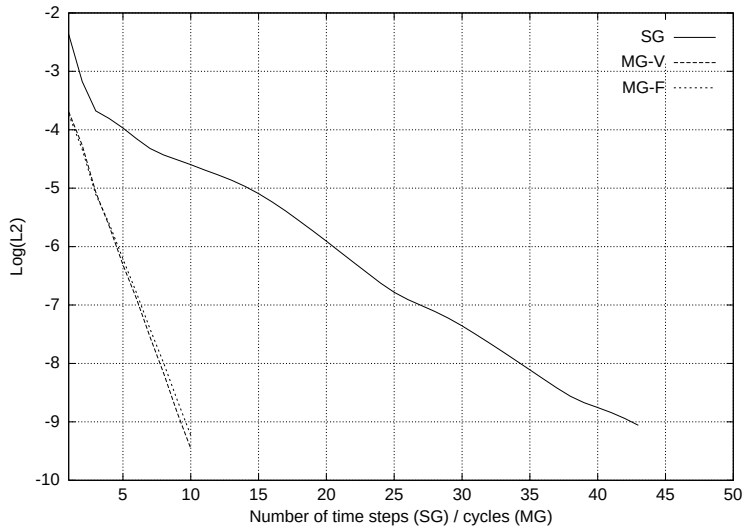


Figure 3.52: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the N-scheme
Convergence of the monogrid (SG) and multigrid (MG-V and MG-F) algorithms

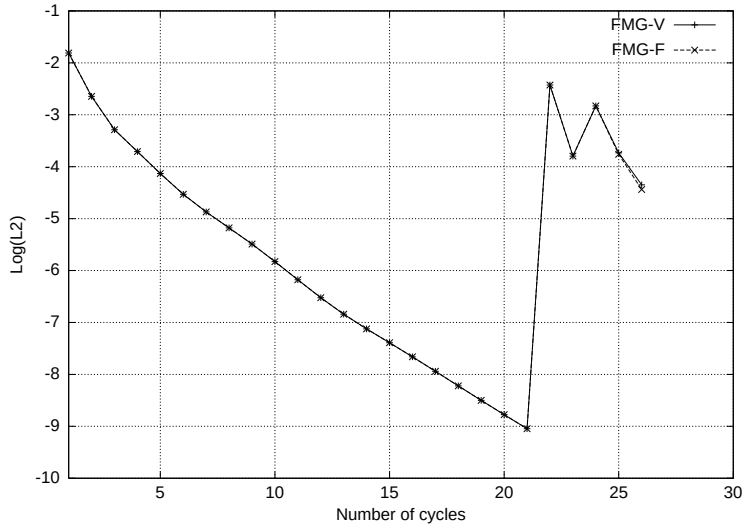


Figure 3.53: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the N-scheme
Convergence of the full multigrid (FMG-V and FMG-F) algorithms

Table 3.22: External flow around an ONERA M6 wing (fully nested meshes)
Spatial approximation based on the N-scheme
Parallel performance results on the SGI Origin 2000

Algorithm	N_p	N_g	# iter (SG) /# cycles (MG)	Wall clock time	$S(N_p)$
SG	4	1	43	4162 s	1.0
	8	1	43	2384 s	1.75
MG-V	4	3	10	4195 s	1.0
	8	3	10	2374 s	1.75
MG-F	8	3	10	2634 s	-
FMG-V	4	3	21/1/2	915 s	1.0
	8	3	21/1/2	503 s	1.8
FMG-F	8	3	21/1/2	540 s	-

has been fixed to 50). At each time step, the resulting linear system is approximately solved using the GMRES method combined to a BMILU preconditioner (the linear threshold has been set to $\varepsilon_l = 10^{-1}$). The non-linear threshold has been fixed to $\varepsilon_{nl} = 10^{-6}$.

- *FAS-MG calculations*. The multigrid hierarchy is composed of the three meshes whose characteristics are given in table 3.21. Two FAS-MG strategies have been considered that are respectively based on a V-cycle and a F-cycle, each of them involving 2 pre- and post-smoothing steps (i.e. 2 backward Euler implicit time steps) on the fine and intermediate grid levels and 4 smoothing steps on the coarsest grid level. However, we only present the results of the application of the V-cycle. Moreover, in the present case, the backward Euler implicit scheme is used in conjunction with a CFL which is varying from one time step to the other (starting value 1, final value 32, multiplication factor 2)

- *Controlled FMG calculations*. We compare the performances of two FMG strategies respectively based on the V-cycle and F-cycle defined previously all parameters being unchanged. In the initial phase of these FMG strategies, the problem defined on the coarsest mesh is solved using the backward Euler implicit scheme with a non-linear threshold fixed to $\varepsilon_{nl} = 10^{-6}$.

On figure 3.54 we compare the convergence profiles associated to the monogrid and FAS-MG strategies while figure 3.55 is dedicated to the FMG strategies. Performance results are given in table 3.23. It is worthwhile to mention that reducing the linear threshold in the resolution of the linear system obtained at each implicit time step (e.g. setting $\varepsilon_l = 10^{-2}$) did not improve the non-linear convergence of all the algorithms. The FMG-V algorithm is 7.3 times faster than the monogrid algorithm on 8 processors. Moreover, the FMG-F algorithm is slightly better than the FMG-V (by a factor 1.2) meaning that the overall gain over the monogrid algorithm is now approaching 8.9 for this test case. We also note that the MG-V algorithm outperforms the monogrid one by a factor 1.9 on 8 processors.

One important concern with the application of the FMG algorithms is related to the accuracy of the resulting solutions. This aspect is partially assessed here through the evaluation of the components of the pressure force acting on the wing:

$$\begin{pmatrix} F_{p,x} \\ F_{p,y} \\ F_{p,z} \end{pmatrix} = \iint_{\Gamma_b} p \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} d\sigma$$

where Γ_b denotes the wing surface and $\vec{n}$ the outward normal vector on Γ_b. Figures 3.56 to 3.61 show the convergence of the X, Y and Z components of the pressure force on the wing (normalized values) for the MG-V algorithm and the two FMG algorithms. The final values of the pressure force components are summarized in table 3.24 showing a good agreement between the computed values.

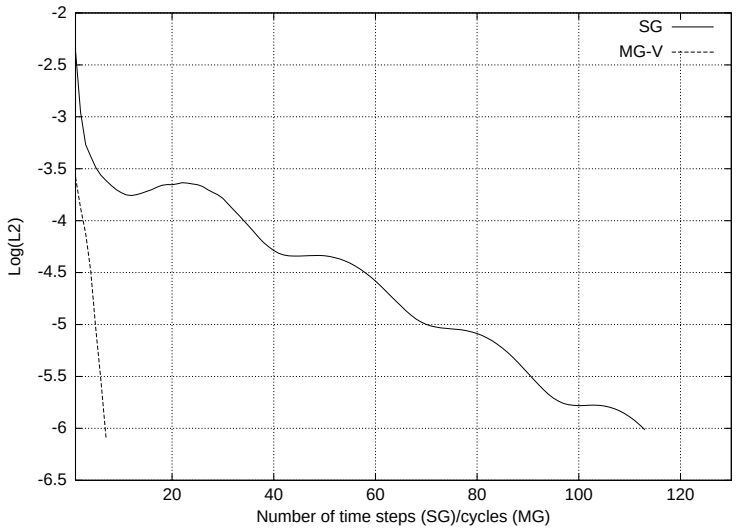


Figure 3.54: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
Convergence of the monogrid (SG) and multigrid (MG-V) algorithms

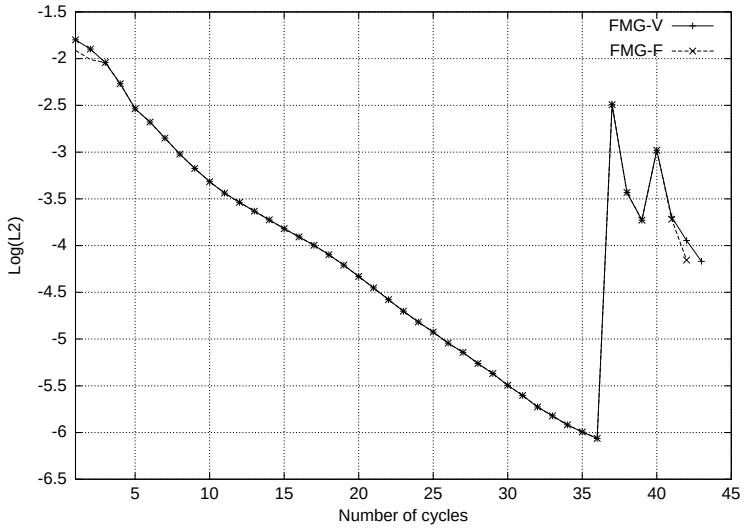


Figure 3.55: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
Convergence of the full multigrid (FMG-V and FMG-F) algorithms

Table 3.23: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
Parallel performance results on the SGI Origin 2000

Algorithm	N_p	N_g	# iter (SG) /# cycles (MG)	Wall clock time	$S(N_p)$
SG	4	1	113	10498 s	1.0
	8	1	113	5034 s	2.0
MG-V	4	3	7	5085 s	1.0
	8	3	7	2648 s	1.9
FMG-V	4	3	36/2/3	1383 s	1.0
	8	3	36/2/3	692 s	2.0
FMG-F	8	3	36/2/2	570 s	-

Table 3.24: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
Final (non-dimensional) values of the pressure force components on the wing

Algorithm	X component	Y component	Z component
MG-V	0.004087	0.158572	-0.057072
FMG-V	0.004117	0.159445	-0.057067
FMG-F	0.004105	0.159166	-0.057062

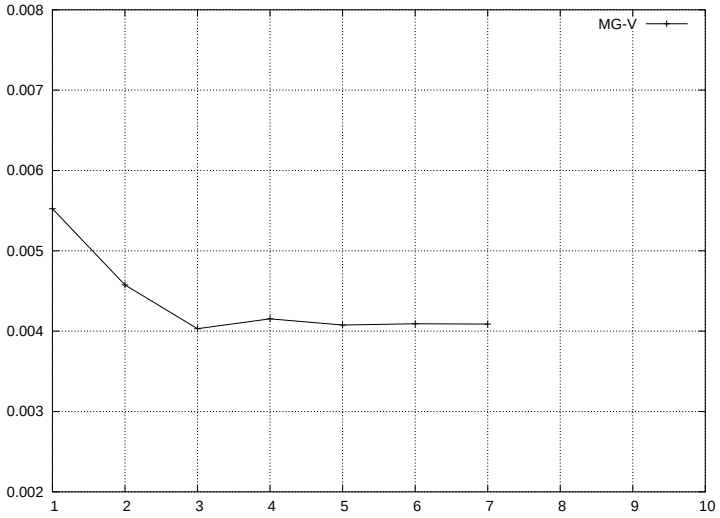


Figure 3.56: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
X pressure force component on the wing (MG-V algorithm)

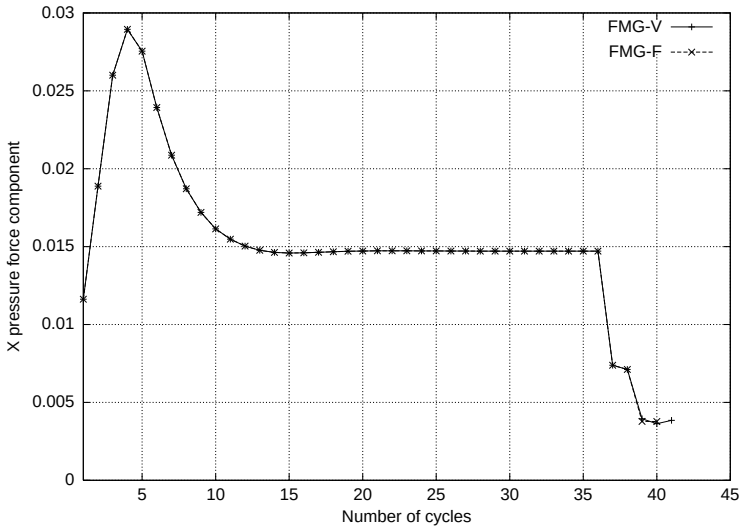


Figure 3.57: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
X pressure force component on the wing (FMG-V and FMG-F algorithms)

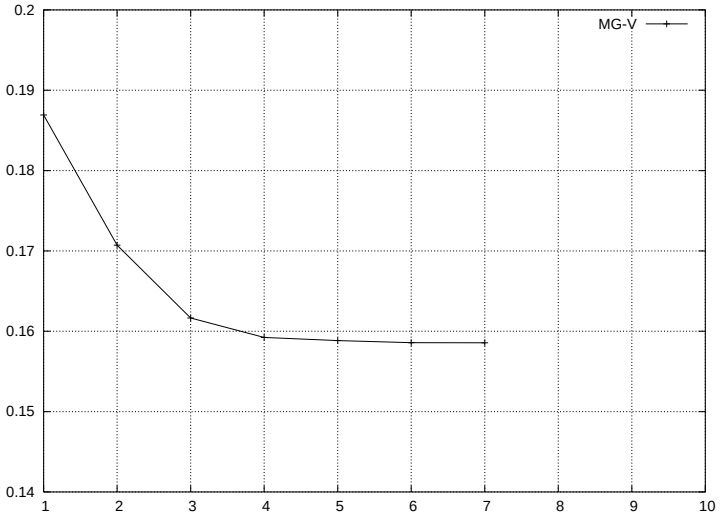


Figure 3.58: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
Y pressure force component on the wing (MG-V algorithm)

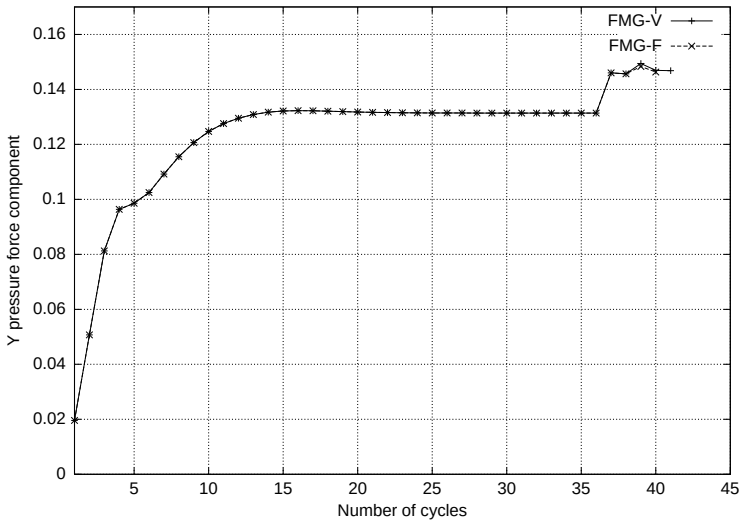


Figure 3.59: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
Y pressure force component on the wing (FMG-V and FMG-F algorithms)

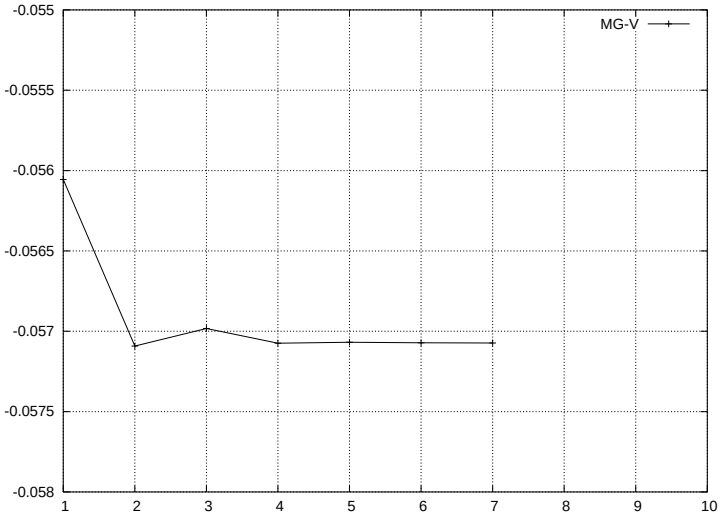


Figure 3.60: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
Z pressure force component on the wing (MG-V algorithm)

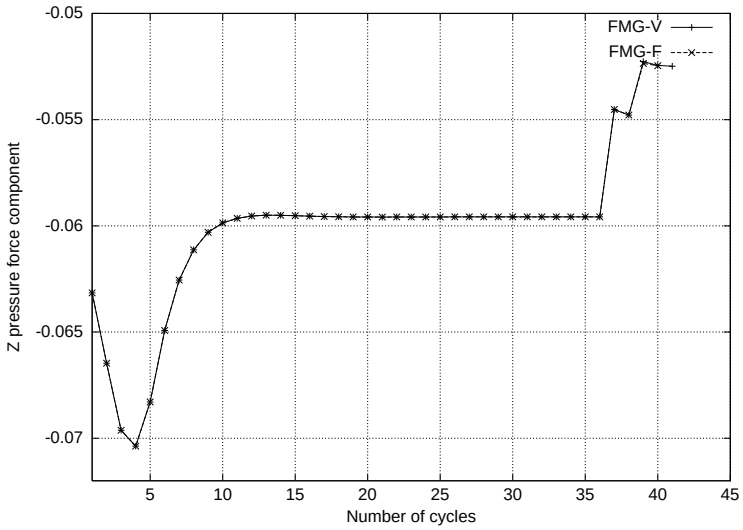


Figure 3.61: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme
Z pressure force component on the wing (FMG-V and FMG-F algorithms)

3.3.6.2.3 Second order PSI-scheme applied to the preconditioned Euler equations. The solution strategies that have been compared are the following:

- *monogrid calculation.* A reference monogrid calculation has been performed on the finest mesh. The semi-discrete equations are time integrated using a linearized backward Euler implicit scheme (the CFL has been fixed to 100). At each time step, the resulting linear system is approximately solved using the GMRES method combined to a BMILU preconditioner (the linear threshold has been set to $\varepsilon_l = 10^{-1}$). The non-linear threshold has been fixed to $\varepsilon_{nl} = 10^{-6}$.
- *FAS-MG calculations.* The multigrid hierarchy is composed of the three meshes whose characteristics are given in table 3.21. Two FAS-MG strategies have been considered that are respectively based on a V-cycle and a F-cycle, each of them involving 2 pre- and post-smoothing steps (i.e. 2 backward Euler implicit time steps) on the fine and intermediate grid levels and 4 smoothing steps on the coarsest grid level. Here, the backward Euler implicit scheme is used in conjunction with a CFL which is varying from one time step to the other (starting value 1, final value 32, multiplication factor 2)
- *Controlled FMG calculations.* We compare the performances of two FMG strategies respectively based on the V-cycle and F-cycle defined previously all parameters being unchanged. In the initial phase of these FMG strategies, the problem defined on the coarsest mesh is solved using the backward Euler implicit scheme with a non-linear threshold fixed to $\varepsilon_{nl} = 10^{-6}$.

On figure 3.62 we compare the convergence profiles associated to the monogrid and FAS-MG strategies while figure 3.63 is dedicated to the FMG strategies. Performance results are given in table 3.23. As previously, we mention that reducing the linear threshold in the resolution of the linear system obtained at each implicit time step (e.g. setting $\varepsilon_l = 10^{-2}$) did not improve the non-linear convergence of all the algorithms. In the present case, the FMG-V and FMG-F algorithms are respectively 3.2 and 2.9 times faster than the monogrid algorithm on 8 processors. The gain bewteen the FMG-V and MG-V algorithm is only by a factor 2.7 (instead of 3.8 for the same algorithms applied to the Euler equations without applying the preconditioning method). Similary, thegain between the MG-V and monogrid algorithm is only by a factor 1.4 (instead of 1.9). This degradation of efficiency of the FAS-MG and FMG algorithms can be attributed to two causes: on one hand, these algorithms are here applied to the overall set of equations with no adaptation to the hyperbolic/elliptic splitting that results from the adopted preconditioning technique; on the other hand, we note that the cost of the initial phase of the FMG algorithms is relatively high (31 iterations of the monogrid algorithm for mesh M1 while the same algorithm requires only 49 iterations on the finest mesh M3 for the same non-linear threshold) suggesting that the preconditioning method does not speedup the convergence on very coarse mesh. Finally, the final values of the pressure force components are summarized in table 3.26.

Table 3.25: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme applied to the preconditioned Euler equations
Parallel performance results on the SGI Origin 2000

Algorithm	N_p	N_g	# iter (SG) /# cycles (MG)	Wall clock time	$S(N_p)$
SG	4	1	49	7933 s	1.0
	8	1	49	4268 s	1.8
MG-V	4	3	5	7056 s	1.0
	8	3	5	3727 s	1.9
MG-F	8	3	7	3019 s	-
FMG-V	4	3	31/2/3	2604 s	1.0
	8	3	31/2/3	1368 s	1.9
FMG-F	8	3	31/2/3	1503 s	-

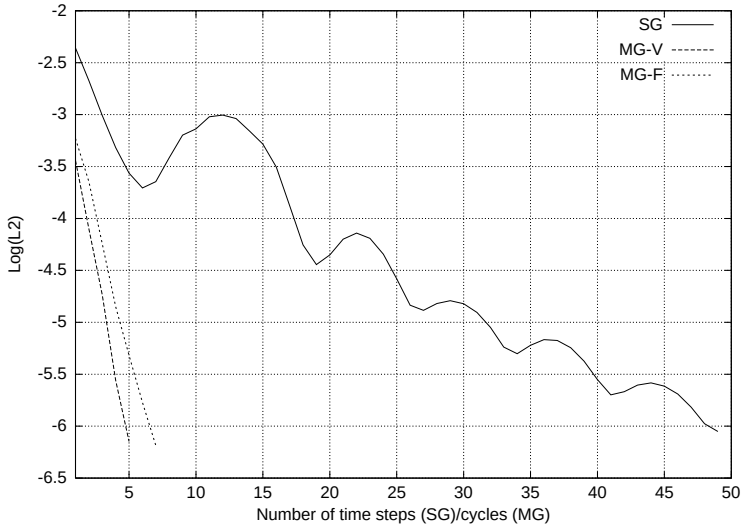


Figure 3.62: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme applied to the preconditioned Euler equations
Convergence of the monogrid (SG) and multigrid (MG-V and MG-F) algorithms

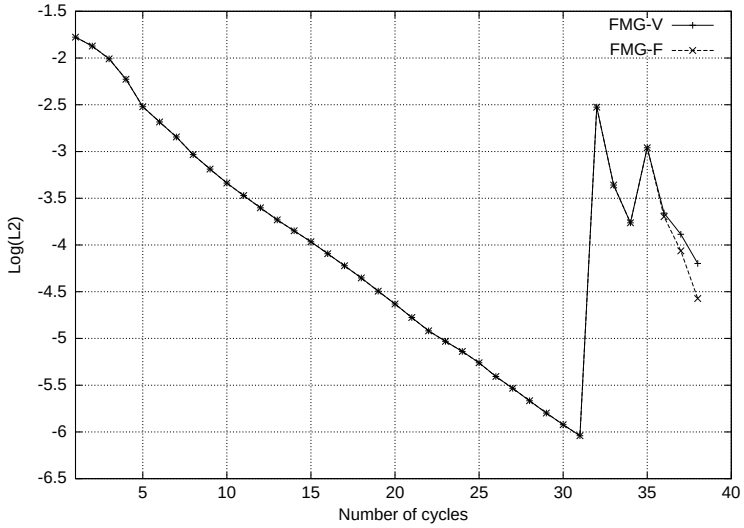


Figure 3.63: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme applied to the preconditioned Euler equations
Convergence of the full multigrid (FMG-V and FMG-F) algorithms

Table 3.26: External flow around an `ONERA M6` wing (fully nested meshes)
Spatial approximation based on the PSI-scheme applied to the preconditioned Euler equations
Final (non-dimensional) values of the pressure force components on the wing

Algorithm	X component	Y component	Z component
MG-V	0.003867	0.159436	-0.057123
MG-F	0.003867	0.159420	-0.057121
FMG-V	0.004002	0.159317	-0.057111
FMG-F	0.003863	0.159453	-0.057117

3.3.6.3 External flow around an ONERA M6 wing: not-nested meshes

We now consider a second series of numerical experiments for the same test case and using a sequence of four meshes whose characteristics are given in table 3.27 below. These meshes have been obtained quasi-independently and are therefore qualified as not-nested.

Table 3.27: Characteristics of not-nested meshes around an ONERA M6 wing
N_V : # vertices - N_T : # tetrahedra - N_F : # boundary faces

Mesh	N_V	N_T	N_F
MU1	5,382	27,199	3,348
MU2	42,305	232,706	15,076
MU3	94,493	555,514	33,126
MU4	316,275	1,940,182	65,486

As in the previous subsection, we report on three series of calculations that have been performed for the following approximation schemes: the first order N-scheme, the second order PSI-scheme and the second order PSI-scheme applied to the preconditioned Euler equations. Steady contour lines of the Mach number on the wing are represented on figures 3.64 to 3.66. As expected, the PSI-scheme applied to the preconditioned Euler equations yields the more accurate solution as can be seen from the resolution of the shock discontinuity.

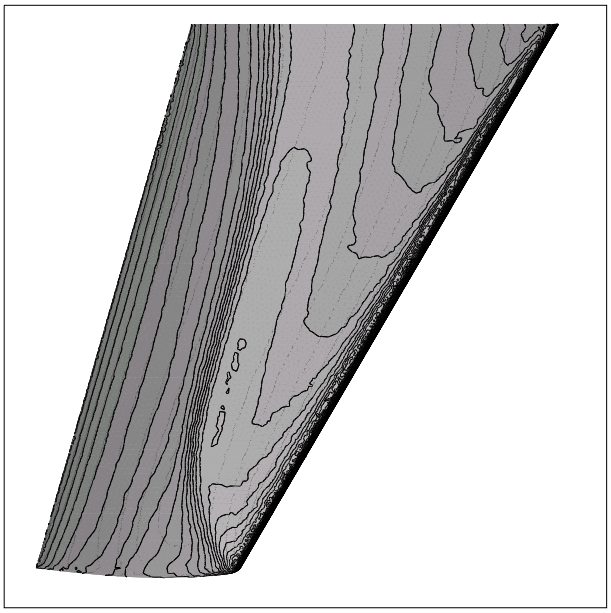


Figure 3.64: External flow around an ONERA M6 wing (not-nested meshes)
Spatial approximation based on the N-scheme
Steady contour lines of the Mach number on the wing

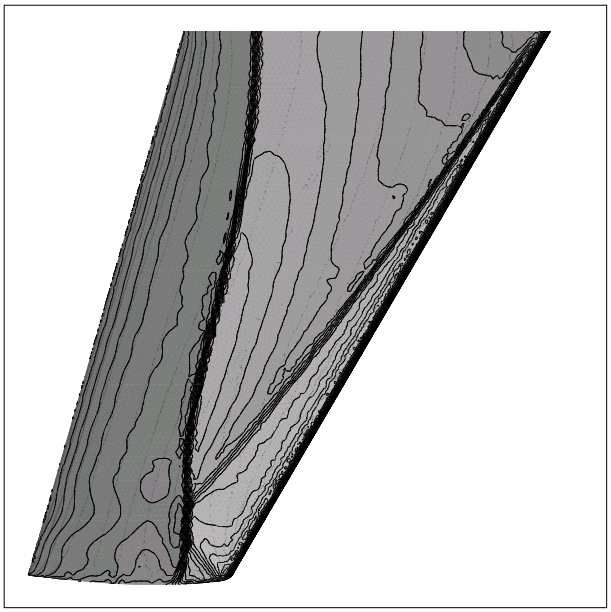


Figure 3.65: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme
Steady contour lines of the Mach number on the wing

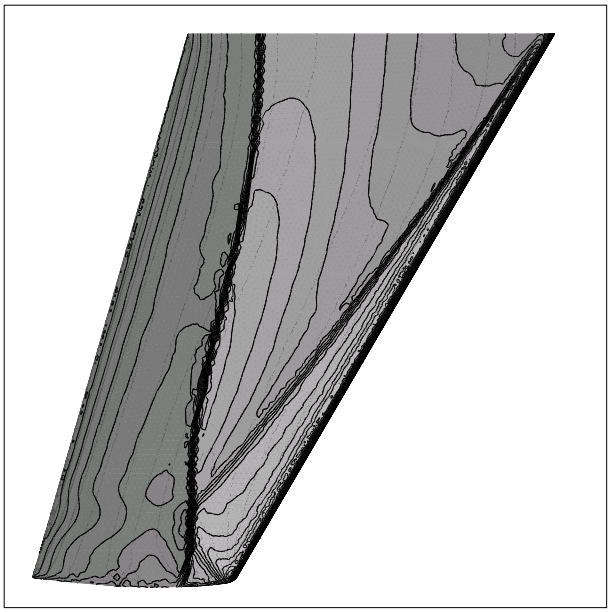


Figure 3.66: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme applied to the preconditioned Euler equations
Steady contour lines of the Mach number on the wing

3.3.6.3.1 Calculations based on the first order N-scheme. Several solution strategies are compared:

- *monogrid calculation.* A reference monogrid calculation has been performed on the finest mesh. The semi-discrete equations are time integrated using a linearized backward Euler implicit scheme (the CFL has been fixed to 100). At each time step, the resulting linear system is approximately solved using the GMRES method combined to a BMILU preconditioner (the linear threshold has been set to $\varepsilon_l = 10^{-1}$). The non-linear threshold has been fixed to $\varepsilon_{nl} = 10^{-9}$.

- *FAS-MG calculations.* The multigrid hierarchy is composed of the four meshes whose characteristics are given in table 3.27. We selected a FAS-MG strategy based on a V-cycle involving 4 pre- and post-smoothing steps on the fine and intermediate grid levels and 4 smoothing steps on the coarsest grid level. As usual, one application of the smoother consists in one time step of the backward Euler implicit scheme adopted for the monogrid calculation.

- *Controlled FMG calculations.* The FMG strategy is based on the previously defined V-cycle all parameters being unchanged. In the initial phase of this FMG strategy, the problem defined on the coarsest mesh is solved using the backward Euler implicit scheme with a non-linear threshold fixed to $\varepsilon_{nl} = 10^{-9}$.

On figure 3.67 we compare the convergence profiles associated to the monogrid and FAS-MG strategies while figure 3.68 is dedicated to the FMG strategies. Performance results are given in table 3.28. As already noticed while using a sequence of fully nested meshes, for this first order calculation, the monogrid algorithm is an efficient solution strategy and the MG-V multigrid algorithms does not result in execution time savings (the MG-V algorithm is even more expensive). The FMG-V strategy is 3.6 times faster than the monogrid algorithm on 16 processors.

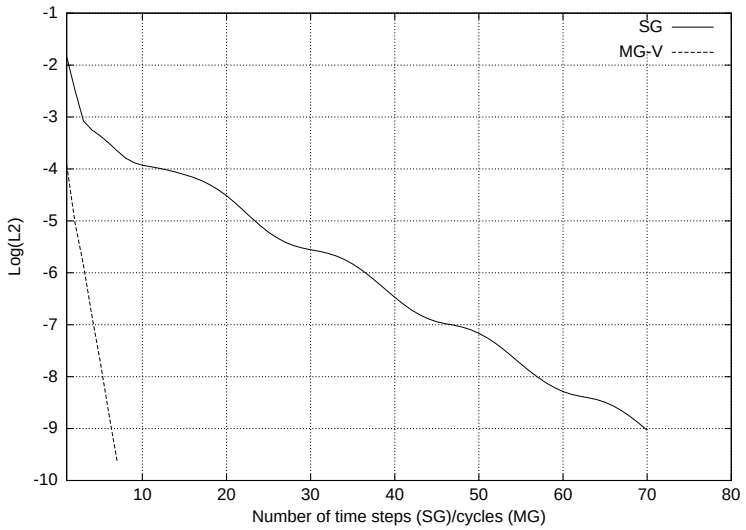


Figure 3.67: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the N-scheme
Convergence of the monogrid (SG) and the multigrid (MG-V) algorithm

3.3.6.3.2 Calculations based on the second order PSI-scheme. The solution strategies that have been considered here are exactly those selected for the calculations based on the N-scheme all parameters being unchanged except for the non-linear threshold that has been fixed to $\varepsilon_{nl} = 10^{-6}$. Convergence profiles are shown on figure 3.69 to 3.71. Performance results are given in table 3.29. This time, the FMG-V algorithm is 7.4 times faster than the monogrid algorithm on 16 processors.

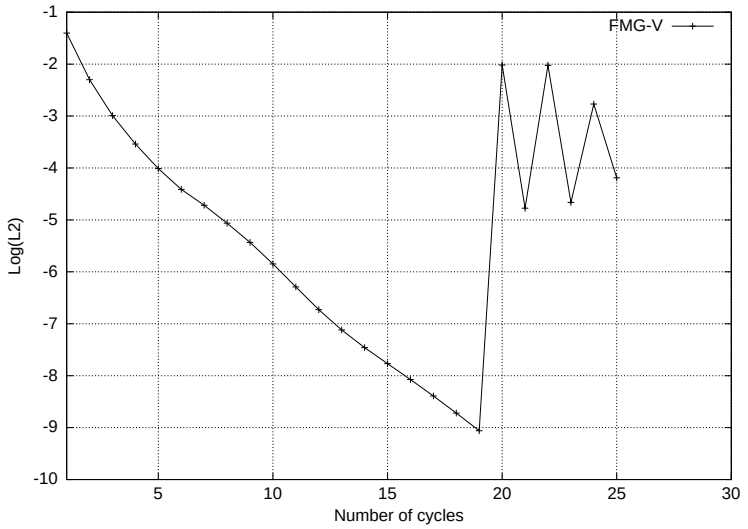


Figure 3.68: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the N-scheme
Convergence of the full multigrid (FMG-V) algorithm

Table 3.28: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the N-scheme
Parallel performance results on the SGI Origin 2000

Algorithm	N_p	N_g	# iter (SG) /# cycles (MG)	Wall clock time
SG	16	1	70	5043 s
MG-V	16	4	7	6020 s
FMG-V	16	4	19/1/1/1	1390 s

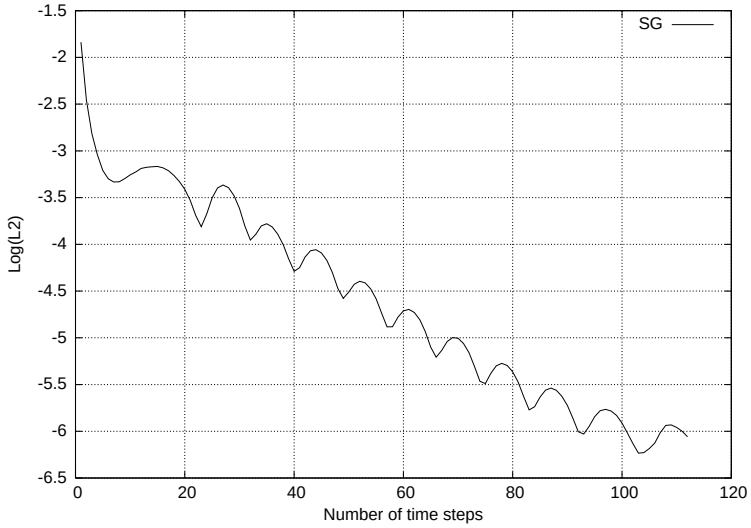


Figure 3.69: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme
Convergence of the monogrid (SG) algorithm

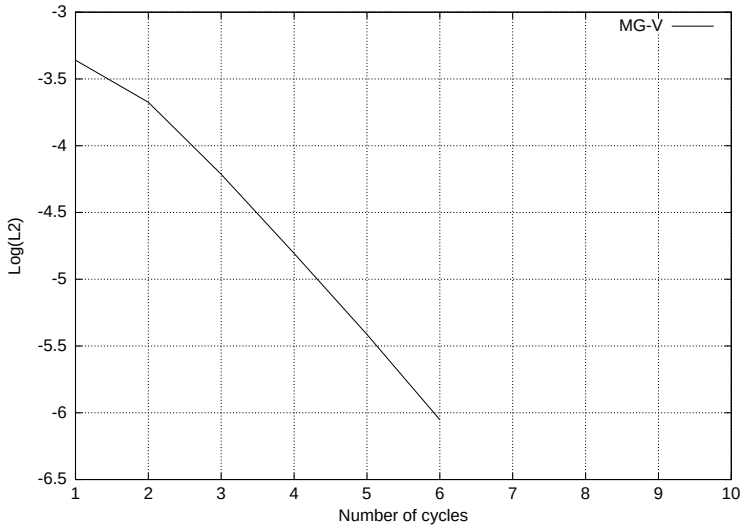


Figure 3.70: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme
Convergence of the multigrid (MG-V) algorithm

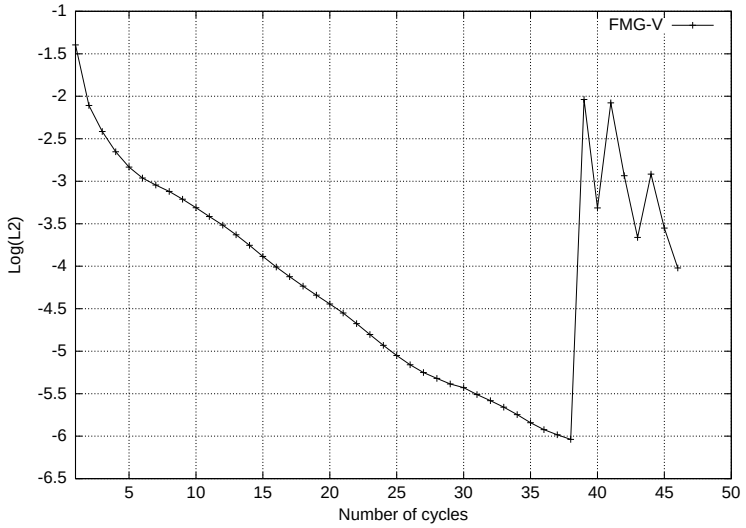


Figure 3.71: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme
Convergence of the full multigrid (FMG-V) algorithm

Table 3.29: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme
Parallel performance results on the SGI Origin 2000

Algorithm	N_p	N_g	# iter (SG) /# cycles (MG)	Wall clock time
SG	16	1	151	9307 s
MG-V	16	4	6	4744 s
FMG-V	16	4	38/1/2/2	1258 s

3.3.6.3.3 Second order PSI-scheme applied to the preconditioned Euler equations. The solution strategies that have been considered are again identical hose selected for the calculations based on the N-scheme except that the CFL has been fixed to 50 for the monogrid calculation (this was found to be the maximum allowable value for this algorithm). Moreover, we only present results for the FMG-V algorithm. The non-linear threshold has been fixed to $\varepsilon_{nl} = 10^{-6}$. Convergence profiles are shown on figure 3.72 and 3.73. Performance results are given in table 3.30. For this calculation, the FMG-V algorithm appears to be 7.6 times faster than the monogrid algorithm on 16 processors. Note that the computing time associated to the monogrid algorithm is approximately twice as high as the corresponding figure obtained when the PSI-scheme is applied to the Euler equations without applying the preconditioning method, despite a slight decrease in the number of time steps. The reason for this behavior is a notable increase in the number of GMRES/BMILU iterations while solving the linear systems resulting from the backward Euler implicit scheme, suggesting that the preconditioning method can have a dramatic impact on the stiffness of the underlying Jacobian matrices.

We conclude this series of results by summarizing in table 3.31 the final values of the X, Y and Z components of the pressure force on the wing (normalized values) for the three schemes.

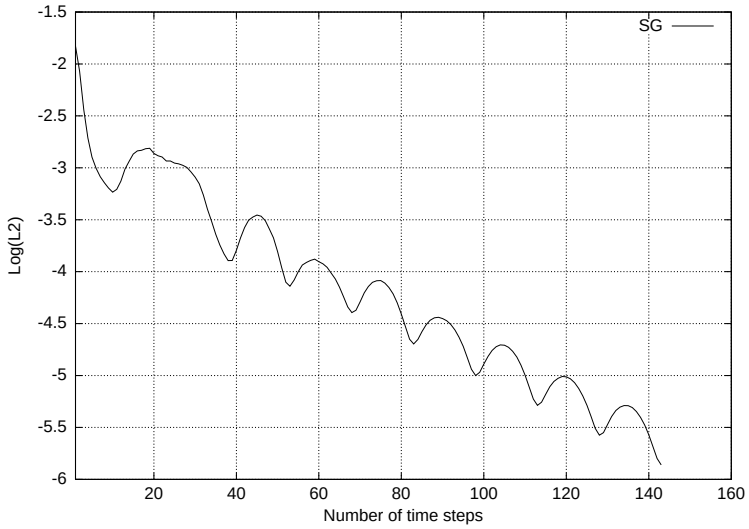


Figure 3.72: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme applied to the preconditioned Euler equations
Convergence of the monogrid (SG) algorithm

Table 3.30: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme applied to the preconditioned Euler equations
Parallel performance results on the SGI Origin 2000

Algorithm	N_p	N_g	# iter (SG) /# cycles (MG)	Wall clock time
SG	16	1	143	18729 s
FMG-V	16	4	32/1/1/1	2470 s

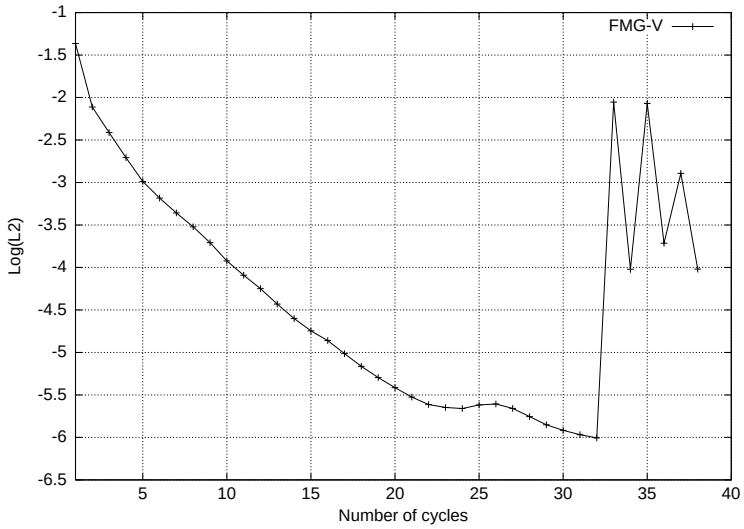


Figure 3.73: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme applied to the preconditioned Euler equations
Convergence of the full multigrid (FMG-V) algorithm

Table 3.31: External flow around an `ONERA M6` wing (not-nested meshes)
Spatial approximation based on the PSI-scheme
Final (non-dimensional) values of the pressure force components on the wing

Scheme	X component	Y component	Z component
N-scheme	7.34225e-03	-5.57764e-02	1.40685e-01
PSI-scheme	3.65002e-05	-5.12173e-02	1.47435e-01
PSI-scheme + prec	-2.73849e-05	-5.13257e-02	1.47583e-01

3.3.6.4 External flow around a Falcon aircraft

The aim of this subsection is to apply the non-linear multi-mesh FAS-MG and FMG algorithms to a flow calculation involving a rather complex geometry. The test case under consideration is the external inviscid subsonic flow around a Falcon aircraft. The free-stream Mach number is set to 0.5 and the angle of attack to 0°. As with the previous test case, the simulation is started from a uniform flow.

A sequence of three meshes has been constructed whose characteristics are given in table 3.32. In the present case, the finest mesh (mesh F1) was constructed first, using the GHS3D tetrahedral mesh generator[63]. Then, meshes F2 and F3 have been obtained by the coarsening method described in subsection 3.3.3.2. Using this method, the resulting meshes are such that vertices of the coarse meshes (i.e. F2 and F3) are a subset of the vertices of the finest mesh (i.e. F1). For this reason, they are qualified as node-nested.

Table 3.32: Characteristics of node-nested meshes around a Falcon aircraft
N_V : # vertices - N_T : # tetrahedra - N_F : # boundary faces

Mesh	N_V	N_T	N_F
F1	7,320	37,964	6,262
F2	58,926	363,124	11,820
F3	455,160	2,647,040	50,208

Calculations have been performed for the first order N-scheme and the second order PSI-scheme. Steady contour lines of the pressure on the aircraft are represented on figures 3.74 to 3.75.

3.3.6.4.1 Calculations based on the first order N-scheme. The following solution strategies are compared:

- *FAS-MG calculation.* The multigrid hierarchy is composed of the three meshes whose characteristics are given in table 3.32. We selected a FAS-MG strategy based on a V-cycle involving 4 pre- and post-smoothing steps on the fine and intermediate grid levels and 4 smoothing steps on the coarsest grid level. As usual, one application of the smoother consists in one time step of the backward Euler implicit scheme adopted for the monogrid calculation. A constant CFL of 50 has been used and the linear solver is GMRES with a BMILU preconditioning (the linear threshold has been set to $\varepsilon_l = 10^{-1}$).

- *Controlled FMG calculations.* The FMG strategy is based on the previously defined V-cycle all parameters being unchanged. In the initial phase of this FMG strategy, the problem defined on the coarsest mesh is solved using the backward Euler implicit scheme with a non-linear threshold fixed to $\varepsilon_{nl} = 10^{-6}$.

Convergence profiles are shown on figures 3.76 and 3.77. Performance results are given in table 3.33. In the present case, the FMG-V algorithm is 14.1 times faster than the MG-V algorithm on 24 processors. Note that the convergence of the MG-V algorithm requires more than 20 cycles which is a relatively high value. In comparison, for the ONERA M6 geometry using the sequence of not-nested meshes, the convergence of the MG-V algorithm (with the same numbers of smoothing steps on each grid level) was obtained in 7 cycles, for the same value of the non-linear threshold. However, in this case, the coarsening ratio characterizing two successive meshes in the multigrid hierarchy is respectively equal to 7.8 (MU1-MU2), 2.2 (MU2-MU3) and 3.4 (MU3-MU4). Recall that the corresponding meshes were obtained independently through several calls to the same mesh generation tool. On the contrary, for the meshes of table 3.32, this ratio is respectively equal to 8.0 (F1-F2) and 7.7 (F2-F3) and was actually taken as a constraint when using the coarsening method described in subsection 3.3.3.2. From the theoretical viewpoint, a coarsening ratio of 8.0 corresponds to the maximum allowable value for an isotropic coarsening strategy. Reducing the coarsening ratio would probably yield a more efficient MG-V algorithm.

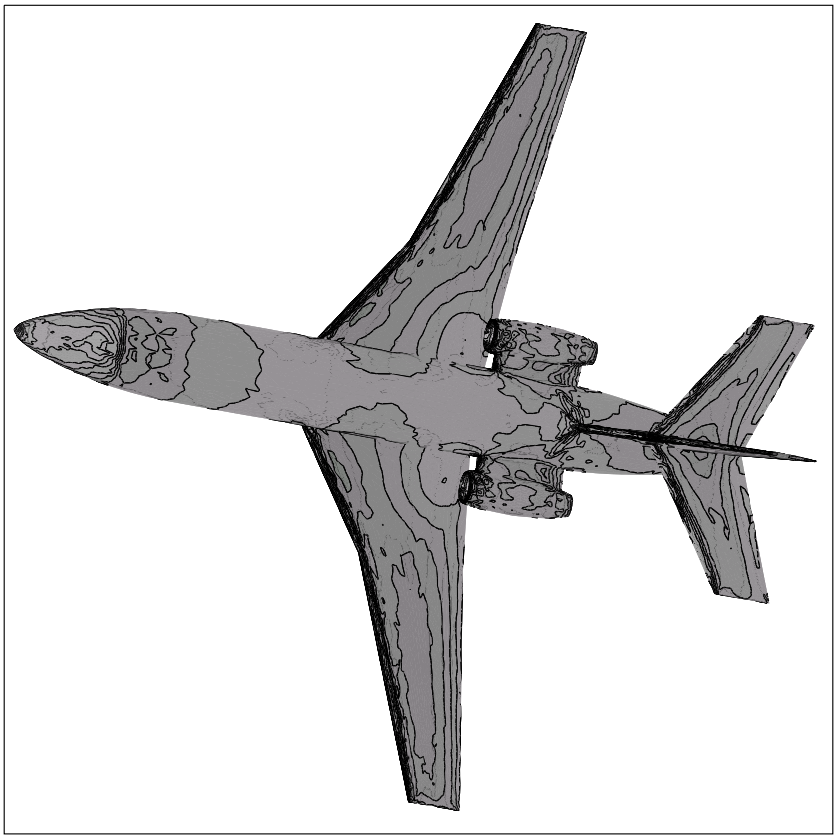


Figure 3.74: External flow around a `Falcon` aircraft (node nested meshes)
Spatial approximation based on the N-scheme
Steady contour lines of the pressure on the aircraft

Table 3.33: External flow around a `Falcon` aircraft (node nested meshes)
Spatial approximation based on the N-scheme
Parallel performance results on the SGI Origin 2000

Algorithm	N_p	N_g	# cycles (MG)	Wall clock time	$S(N_p)$
MG-V	12	3	21	23827 s	1.0
	24	3	21	13004 s	1.85
FMG-V	12	3	50/1/1	1594 s	1.0
	24	3	50/1/1	919 s	1.75

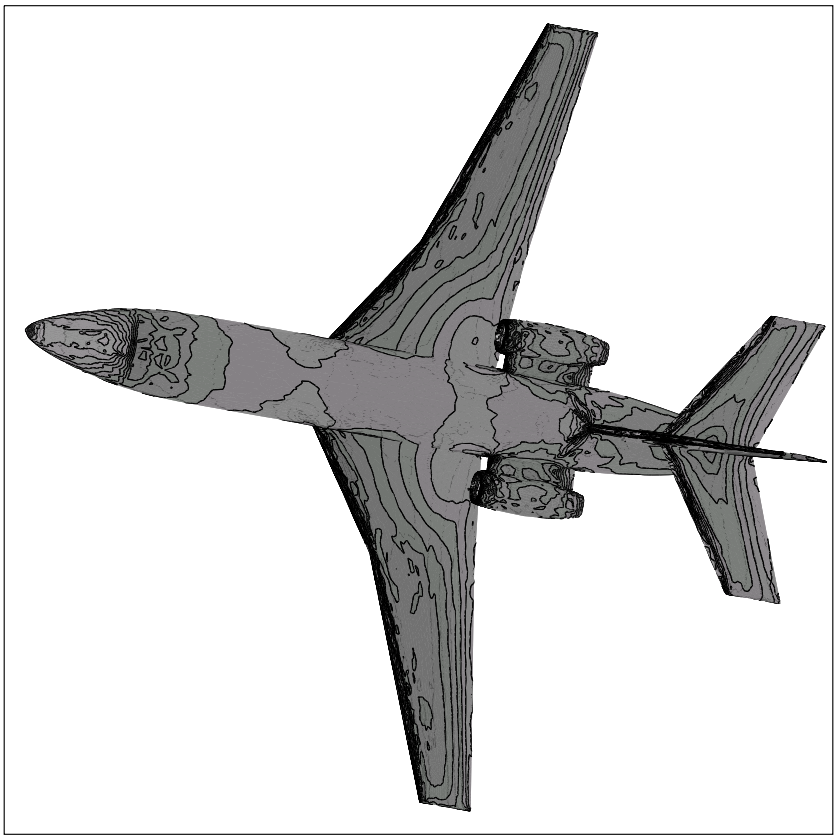


Figure 3.75: External flow around a `Falcon` aircraft (node nested meshes)
Spatial approximation based on the PSI-scheme
Steady contour lines of the pressure on the aircraft

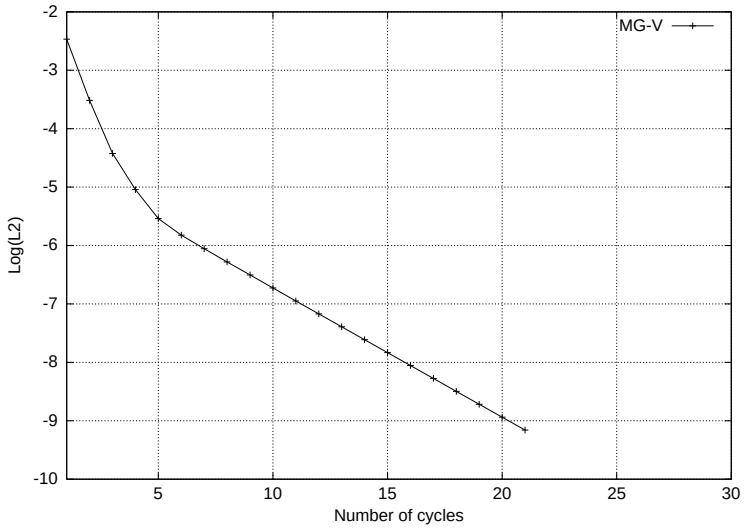


Figure 3.76: External flow around a `Falcon` aircraft (node nested meshes)
Spatial approximation based on the N-scheme
Convergence of the multigrid (MG-V) algorithm

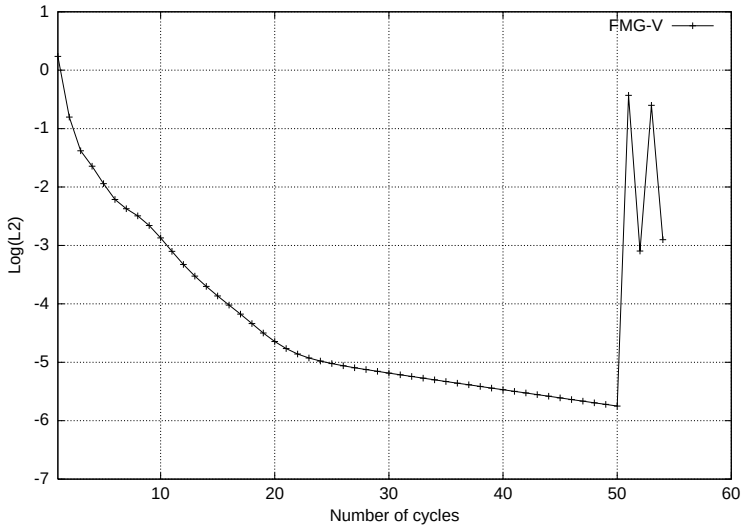


Figure 3.77: External flow around a `Falcon` aircraft (node nested meshes)
Spatial approximation based on the N-scheme
Convergence of the full multigrid (FMG-V) algorithm

3.3.6.4.2 Calculations based on the second order PSI-scheme. The solution strategies that have been considered here are exactly those selected for the calculations based on the N-scheme all parameters being unchanged except for the CFL that has been set to 10 and the non-linear threshold that has been fixed to $\varepsilon_{nl} = 10^{-6}$. Convergence profiles are shown on figures3.78 and 3.79. Performance results are given in table 3.34. Note that in the initial phase of the FMG calculation, the required non-linear threshold has not been obtained within 100 time steps (this value has been set for the maximun number of time steps that could be performed in this phase). As a result, the FMG-V algorithm is only 7.5 times faster than the MG-V algorithm on 24 processors. It is worthwhile to mention that and ideal parallel speedup is obtained for the MG-V algorithm. For the FMG algorithm, a slight degradation is observed which is the result of the reltively high number of time steps in the initial phase which is involving a very coarse mesh f

Table 3.34: External flow around a `Falcon` aircraft (node nested meshes)
Spatial approximation based on the PSI-scheme
Parallel performance results on the SGI Origin 2000

Algorithm	N_p	N_g	# cycles (MG)	Wall clock time	$S(N_p)$
MG-V	12	3	21	22394 s	1.0
	24	3	21	11140 s	2.0
FMG-V	12	3	100/1/2	2737 s	1.0
	24	3	100/1/2	1478 s	1.85

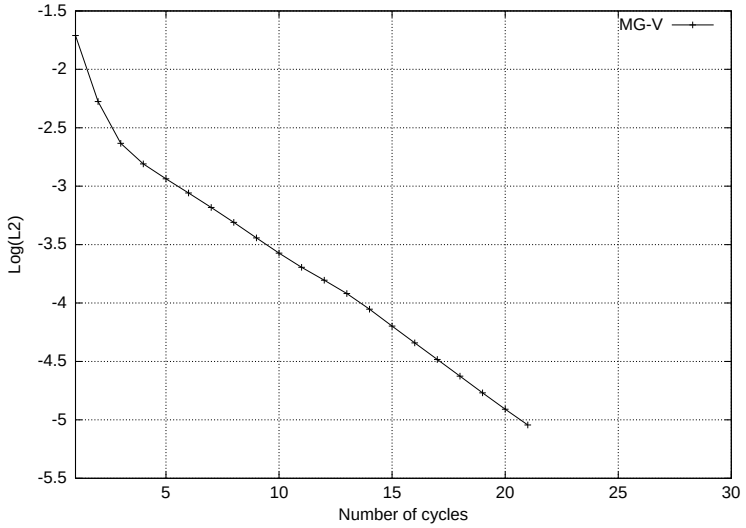


Figure 3.78: External flow around a `Falcon` aircraft (node nested meshes)
Spatial approximation based on the PSI-scheme
Convergence of the multigrid (MG-V) algorithm

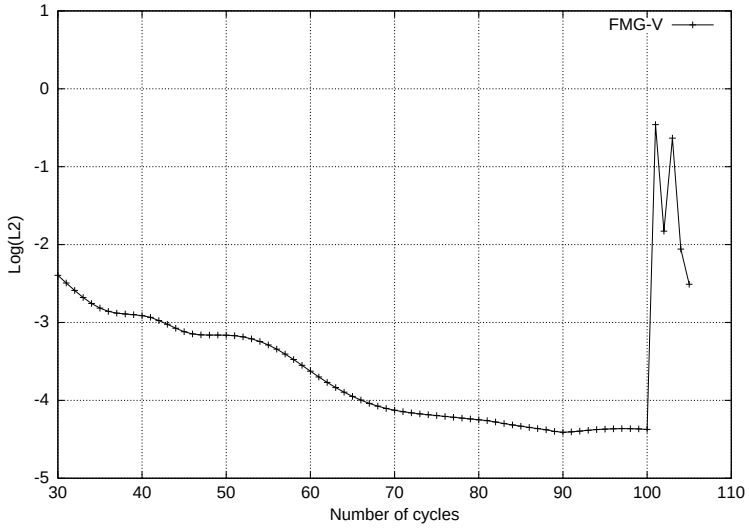


Figure 3.79: External flow around a `Falcon` aircraft (node nested meshes)
Spatial approximation based on the PSI-scheme
Convergence of the full multigrid (FMG-V) algorithm

3.3.7 Conclusion

In this section, we have described our contributions concerning the development of parallel non-linear multigrid algorithms for the acceleration of three-dimensional compressible flows on unstructured tetrahedral meshes. The approach considered here is an alternative to the multigrid by volume agglomeration method studied in section 3.1. Its main features are the following:

- contrary to what is done in section 3.1, the multigrid algorithms directly act on the non-linear, space discretized equations. In this context, the basic multigrid algorithm is the Full Approximation Scheme (FAS). The time stepping scheme (either explicit or implicit) plays the role of the smoother. Both standard V-, F- and W-cycle based MG algorithms, as well as the Full Multigrid (FMG) algorithm, have been studied.
- the formulation of the multigrid algorithms rely on a multi-mesh hierarchy. Each element of this hierarchy is a tetrahedral mesh τ_k corresponding to the discretization of the same geometry. For inviscid flow calculations, two successive meshes of this hierarchy are characterized by a coarsening ratio $\frac{h_{k+1}}{h_k} = 2$ where h_k is the maximal length of the edges of τ_k. Such meshes can be obtained using different strategies: (1) by successive refinement of a very coarse mesh, (2) through a series of independent mesh generation steps or, (3) with the help of an appropriate (automatic) coarsening tool. In this study, no assumption has been made regarding the way meshes are nested in the multi-mesh hierarchy. This means that a specific localization tools has been developed for obtaining the data needed for implementing the inter-grid transfer operators. Moreover, an original algorithm has been designed for coarsening tetrahedral meshes while respecting the boundary of the underlying geometry.

The main advantages of the present approach are: (1) its simplicity of implementation (the FAS-MG and FMG algorithms are built on top of the existing monogrid solver) and (2) the same space approximation method can be applied on each (finite element) grid level. It is also worthwhile to mention that the developed multigrid kernels (inter-grid transfer operators, FAS-MG cycles and FMG algorithm) are essentially independent of the numerical framework characterizing the starting point monogrid solver. Thus, they can easily be reused in another application context.

Chapter 4

Domain decomposition methods for compressible flows

The objective of this chapter is to present our contributions concerning the design of domain decomposition methods for the calculation of compressible flows modeled by the system of Euler (case of an inviscid fluid) or Navier-Stokes (case of a viscous fluid) equations. In both cases, the methods that we propose rely on the formulation of an additive Schwarz algorithm involving transmission conditions that are derived from a weak formulation of the underlying boundary value problem. In the case of inviscid flows, these interface conditions are equivalent to requiring the continuity of the components of the normal convective flux. For laminar viscous flows, the interface conditions generalize the one used for the Euler conditions by imposing also the continuity of the components of the normal viscous flux. We note that similar formulations were previously studied by Quarteroni and Stolcis[116].

Section 4.1 is dedicated to the calculation of inviscid flows while section 4.2 concentrates on the solution of the Navier-Stokes equations for the calculation of steady and unsteady laminar viscous flows. In section 4.1, the proposed method is studied from both the theoretical and numerical points of view. For what concern the calculation of viscous flows, we only only discuss the numerical aspects of the application of the methodology to the solution of the Navier-Stokes equations.

4.1 Domain decomposition for inviscid flows

4.1.1 Introduction

As mentioned above, our approach to domain decomposition solution of inviscid flows relies on the formulation of an additive Schwarz algorithm involving transmission (or interface) conditions that are naturally derived from a weak formulation of the underlying boundary value problem. These interface conditions (from now on referred as *natural* or *classical* interface conditions) amount to impose the continuity of the normal convective flux. Nevertheless, we will show that well posed local boundary value problems are obtained by simply using interface conditions that are Dirichlet conditions for the characteristic variables corresponding to incoming waves, thus taking into account the hyperbolic nature of the Euler equations. We note that similar formulations were previously studied by Bjørhus[11] for the semi-discrete Euler equations and by Quarteroni and Stolcis[116] for more complicated models (e.g. the Navier-Stokes equations for viscous flows).

When dealing with supersonic flows, whatever the space dimension is, imposing the appropriate characteristic variables as interface conditions leads to a convergence of the algorithm which is optimal with respect to the number of subdomains. This property is generally lost for subsonic flows except for the case of one-dimensional problems, when the optimality is again expressed as the number of iterations being equal to the number of subdomains (see Bjørhus [11] and Quarteroni [115] for more details). For higher space dimensions, one cannot

analyze the convergence of the algorithm in the same way. Therefore, a new kind of approach is required in the latter case. In a similar context, Clerc [33] gives a convergence proof for the additive Schwarz algorithm applied to the solution of a general linear hyperbolic system of PDEs, in the two- and three-dimensional cases, based on an energy estimate of the error vector. We note that this proof is limited to a non-overlapping decomposition of the computational domain and no quantitative results on the convergence rate are provided.

In section 4.1.2, we outline the basic principles for the formulation of non-overlapping domain decomposition algorithms for hyperbolic systems of partial differential equations. The proposed framework is greatly inspired from Gastaldi and Gastaldi[61], Gastaldi *et al.*[62], Quarteroni and Stolcis[116], Quarteroni and Valli[117] and Nataf[107]. We also refer to Smith *et al.*[130] and Quarteroni and Valli[119] for a general introduction, as well as an in-depth description of domain decomposition algorithms. Most of the discussion here is undertaken in the context of a general linear hyperbolic system. Then, in section 4.1.3, we apply the proposed methodology to the solution of the Euler equations and we study the convergence of the additive Schwarz algorithm in the two- and three-dimensional case, for overlapping and non-overlapping decompositions, by applying a Fourier analysis. For the sake of simplicity, we limit the analysis to two- and three-subdomain decompositions and we provide analytical expressions of the convergence rate of the Schwarz algorithm applied to the linearized equations. Surprisingly, there exists flow conditions for which the asymptotic convergence rate is equal to zero. In this section, we also present results of numerical experiments that aim at validating the theoretical analysis. Finally, in section 4.1.4, we apply the proposed algorithm to the calculation of two-dimensional steady inviscid compressible flows.

4.1.2 Additive Schwarz algorithm for a general hyperbolic system

Here, we briefly review the main definitions and properties of hyperbolic systems of conservation laws that are of interest to our study. Then we introduce an additive Schwarz algorithm which is based on transmission conditions at subdomain interfaces that take into account the hyperbolic nature of the problem. In addition, we recall some existing results concerning the convergence of the algorithm.

We consider a general system of hyperbolic conservation laws of the form :

$$\frac{\partial W}{\partial t} + \mathrm{div}(\vec{\mathbb{F}}(W)) = \frac{\partial W}{\partial t} + \sum_{i=1}^{d} \frac{\partial F_i(W)}{\partial x_i} = 0 \quad \text{with} \quad W \in \mathbb{R}^p \tag{4.1}$$

where d denotes the space dimension and p the dimension of the system. The flux functions F_i are assumed differentiable with respect to the state vector $W = W(x,t)$. In the general case, the flux functions are non-linear functions of W. Under the assumption that the solution $W = W(x,t)$ is regular, we can also write a non-conservative (or quasi-linear) equivalent form of equation (4.1) :

$$\frac{\partial W}{\partial t} + \sum_{i=1}^{d} A_i(W) \frac{\partial W}{\partial x_i} = 0 \tag{4.2}$$

where the $A_i(W) = \dfrac{\partial F_i}{\partial W}(W)$ are the Jacobian matrices of the flux vectors. Recall that system (4.1) is said to be hyperbolic if, for any unitary real vector $\vec{n} \in \mathbb{R}^d$, the matrix $A_n(W) = \displaystyle\sum_{i=1}^{d} n_i A_i(W)$ is diagonalizable with real eingenvalues. We are particularly interested in the situation where system (4.1) is integrated in time using a backward Euler implicit scheme involving a linearization of the flux functions. In that case we have :

$$\frac{\delta W}{\Delta t} + \sum_{i=1}^{d} \frac{\partial}{\partial x_i} \left[\frac{\partial F_i}{\partial W}(\bar{W}) \delta W \right] = -\mathrm{div}(\vec{\mathbb{F}}(\bar{W})) \tag{4.3}$$

where $\delta W = W - \bar{W}$. When δW is assumed regular, we can write the non-conservative form of system (4.3) :

$$\left[\frac{1}{\Delta t}\mathsf{Id}+\sum_{i=1}^{d}\frac{\partial}{\partial x_i}\left[\frac{\partial F_i}{\partial W}(\bar{W})\right]\right]\delta W+\sum_{i=1}^{d}\left[\frac{\partial F_i}{\partial W}(\bar{W})\right]\frac{\partial \delta W}{\partial x_i}=-\mathsf{div}(\vec{\mathbb{F}}(\bar{W})) \tag{4.4}$$

System (4.4) can be symmetrized through the multiplication by an operator Σ (see for example Barth[8]) which, for hyperbolic systems admitting an entropy function, is given by the Hessian matrix of this entropy. This operation results in the following first order system :

$$A_0(\bar{W})\delta W+\sum_{i=1}^{d}A_i(\bar{W})\frac{\partial \delta W}{\partial x_i}=f \tag{4.5}$$

with :

$$\left\{\begin{array}{ll} A_0(\bar{W}) & =\Sigma\left[\frac{1}{\Delta t}\mathsf{Id}+\sum_{i=1}^{d}\frac{\partial \delta}{\partial x_i}\left[\frac{\partial F_i}{\partial W}(\bar{W})\right]\right] \\ A_i(\bar{W}) & =\Sigma\left[\frac{\partial F_i}{\partial W}(\bar{W})\right] \\ f & =-\Sigma\mathsf{div}(\vec{\mathbb{F}}(\bar{W})) \end{array}\right. \tag{4.6}$$

Now, let $\vec{n}$ denote the outward normal vector to $\partial\Omega$. From now on, we simply note A_n instead of $A_n(\bar{W})$. Moreover, A_nW will denote the normal trace of W on $\partial\Omega$. When dealing with boundary conditions for a hyperbolic system of PDEs, it is well known that one cannot impose all the components of W on the boundary $\partial\Omega$. Instead, the direction of propagation of the information has to be taken into account in order to obtain a well posed boundary value problem (BVP) for system (4.5). More precisely, the number and type of boundary conditions that must be imposed on $\partial\Omega$ are deduced from the expression of system (4.5) in terms of characteristic variables and is related to information entering the domain Ω. A more rigorous discussion of boundary conditions treatment for hyperbolic systems from gas dynamics, in terms of characteristic variables, is for example given in [64] (see also Quarteroni and Valli[119] for a discussion in the context of domain decomposition algorithms). Using the diagonalization of A_n which writes as $A_n=T\Lambda_nT^{-1}$ we have :

$$\left\{\begin{array}{c} A_n^{\pm}=T\Lambda_n^{\pm}T^{-1} \\ \Lambda_n^{\pm}=\mathsf{diag}(\lambda_i^{\pm})_{1\leq i\leq p} \ \text{ with } \ \lambda_i^{\pm}=\frac{1}{2}(\lambda_i\pm|\lambda_i|) \\ \text{and with } \ A_n^{+}W=-A_{-n}^{-}W \end{array}\right.$$

In order to obtain a well posed BVP, we have to impose boundary conditions of the form :

$$A_n^{-}W=A_n^{-}g$$

where A_n^{-} is used to select the information entering the domain Ω. We recall below a well known result concerning the boundary value problem associated to system (4.5)-(4.6) that can be found in [32] :

Theorem 1 *Assume $f\in L^2(\Omega)^p$ and $g\in L^2_A(\partial\Omega)$ with :*

$$\begin{aligned} L^2_A(\partial\Omega) &= \{W \textit{ such that } \int_{\partial\Omega}|A_n|W\cdot W d\sigma<\infty\} \\ L^2_{1/A}(\partial\Omega) &= \{W \textit{ such that } \int_{\partial\Omega}|A_n|^{-1}W\cdot W d\sigma<\infty\} \end{aligned}$$

the following BVP problem is well posed :

$$\left\{ \begin{array}{rcl} \mathcal{L}(W) = A_0 W + \sum_{i=1}^{d} A_i \dfrac{\partial W}{\partial x_i} & = & f \ \ \textit{in} \ \ \Omega \\ A_n^- W & = & A_n^- g \ \ \textit{on} \ \ \partial\Omega \end{array} \right. \tag{4.7}$$

where the unique solution W of (4.7) lies in the space $\tilde{H}$ with :

$$\tilde{H} = \{W \in L^2(\Omega)^p \ \textit{such that} \ \sum_{i=1}^{d} A_i \frac{\partial W}{\partial x_i} \in L^2(\Omega)^p \ \textit{and} \ W|_{\partial\Omega} \in L^2_A(\partial\Omega)\}$$

Furthermore, if $f = 0$ we have the estimate :

$$C_0 \|W\|^2_{L^2(\Omega)} + \|A_n^+ W\|^2_{L^2_{1/A}(\partial\Omega)} \leq \|A_n^- g\|^2_{L^2_{1/A}(\partial\Omega)}$$

In the following, we are interested in solving the BVP problem (4.7) by an additive Schwarz algorithm based on transmission conditions at subdomain interfaces that consist in Dirichlet conditions for the characteristic variables corresponding to incoming waves (a formulation already considered by Quarteroni and Stolcis[116]). In other words, the treatment of the condition on the physical boundary in eq. (4.7) will be extended to the artificial boundaries defined by interfaces between neighboring subdomains.

We consider a decomposition of the domain Ω into N overlapping or non-overlapping subdomains $\bar{\Omega} = \bigcup_{i=1}^{N} \bar{\Omega}_i$. We denote by $\vec{n}_{ij}$ the normal vector at any point of the interface between Ω_i and a neighboring subdomain Ω_j directed from Ω_i to Ω_j. The domain decomposition approach for solving (4.7) consists in defining well posed subproblems so that a local solution on a given subdomain Ω_i is the restriction of the global solution on Ω to Ω_i. The subproblems will inherit the physical boundary conditions of the global problem for the part of $\partial\Omega_i$ which intersects $\partial\Omega$; in addition, appropriate interface conditions are added to the definition of the subproblems for the part of $\partial\Omega_i$ which is common to its neighboring subdomains:

$$\left\{ \begin{array}{l} \mathcal{L}(W_i) = f|_{\Omega_i} = f_i \\ A_n^- W_i = A_n^- g \ \ \text{on} \ \ \partial\Omega_i \cap \partial\Omega \\ \text{Interface conditions on} \ \ \Gamma_{ij} = \partial\Omega_i \cap \overline{\Omega}_j \ \ \text{for} \ \ \Omega_i \cap \Omega_j \neq \emptyset \end{array} \right. \tag{4.8}$$

Assume that local solution W_i is prolongated by zero on Ω/Ω_i; then a necessary and sufficient condition to insure that $\sum_{i=1}^{N} W_i$ is the solution of the global problem (4.7) is that on Γ_{ij} the following conditions are verified :

$$A^-_{n_{ij}} W_i + A^+_{n_{ji}} W_j = 0 \ \ \text{and} \ \ A^+_{n_{ij}} W_i + A^-_{n_{ji}} W_j = 0 \tag{4.9}$$

This result can be deduced from the variational formulation of problem (4.9). For simplicity, we consider the two-subdomain case $\Omega = \Omega_1 \cup \Omega_2$. Let $\vec{n}_{\partial\Omega}$ denote the outward normal on $\partial\Omega$ and $\vec{n} = \vec{n}_1 = -\vec{n}_2$ the outward normal on Γ_{12} (directed from Ω_1 to Ω_2), and let $W = W_1 + W_2$ and $f = f_1 + f_2$. We introduce variational formulations which are based on the following space of test functions :

$$V = \{X \in L^2(\Omega)^p \ \ \text{such that} \ \ \ \sum_{i=1}^{d} A_i X \in H(\text{div}, \Omega)^p, \quad A_n X \in L^2_A(\Gamma)\}$$

On one hand, we integrate by parts on the global domain Ω and, on the other hand, we integrate by parts on each subdomain Ω_1 and Ω_2 (the notation ∂_{x_k} is used to denote $\dfrac{\partial}{\partial x_k}$) :

$$\begin{aligned}
\int_\Omega [A_0 W + \sum_{k=1}^{d} A_k \partial_{x_k} W] X d\omega &= \int_\Omega [A_0^T X - \sum_{k=1}^{d} \partial_{x_k}(A_k X)] W d\omega + \int_{\partial\Omega} (A_{n_{\partial\Omega}} W) X d\gamma \\
&= \int_{\Omega_1} [A_0^T X - \sum_{k=1}^{d} \partial_{x_k}(A_k X)] W_1 d\omega + \int_{\partial\Omega_1 \cap \partial\Omega} (A_{n_{\partial\Omega}} W_1) X d\gamma \\
&+ \int_{\Omega_2} [A_0^T X - \sum_{k=1}^{d} \partial_{x_k}(A_k X)] W_2 d\omega + \int_{\partial\Omega_2 \cap \partial\Omega} (A_{n_{\partial\Omega}} W_2) X d\gamma \\
&= \int_{\Omega_1} [A_0 W_1 + \sum_{k=1}^{d} A_k \partial_{x_k} W_1] X d\omega - \int_{\Gamma_{12}} (A_{n_1} W_1) X d\gamma \\
&+ \int_{\Omega_2} [A_0 W_2 + \sum_{k=1}^{d} A_k \partial_{x_k} W_2] X d\omega - \int_{\Gamma_{12}} (A_{n_2} W_2) X d\gamma \\
&= \int_{\Omega_1} [A_0 W_1 + \sum_{k=1}^{d} A_k \partial_{x_k} W_1] X d\omega \\
&+ \int_{\Omega_2} [A_0 W_2 + \sum_{k=1}^{d} A_k \partial_{x_k} W_2] X d\omega + \int_{\Gamma_{12}} [A_n W_2 - A_n W_1] X d\gamma
\end{aligned}$$

If W_1 and W_2 are the (local) solutions of (4.8) then we must have :

$$\int_\Gamma [A_n W_1 - A_n W_2] X d\gamma = 0 \quad \forall X \in V$$

Therefore a necessary and sufficient condition to insure that $W_1 + W_2$ is the (global) solution of (4.7) is :

$$A_n W_1 = A_n W_2 \quad \text{on} \quad \Gamma_{12} \tag{4.10}$$

Since A_n is non-singular then (4.10) implies that :

$$W_1 = W_2 \quad \text{on} \quad \Gamma_{12} \tag{4.11}$$

therefore :

$$\left(T \Lambda_n^{\pm} T^{-1}\right) W_1 = \left(T \Lambda_n^{\pm} T^{-1}\right) W_2 \quad \text{on} \quad \Gamma_{12} \tag{4.12}$$

which yields :

$$A_n^- W_1 = A_n^- W_2 \quad \text{and} \quad A_n^+ W_1 = A_n^+ W_2 \tag{4.13}$$

Conversely, since $A_n = A_n^+ + A_n^-$, conditions (4.13) imply (4.12), which concludes the proof.

Let $W_i^{(0)}$ denote the initial approximation of the solution in subdomain Ω_i. A general formulation of an additive Schwarz algorithm for computing $(W_i^{(k+1)})_{1 \le i \le N}$ from $(W_i^{(k)})_{1 \le i \le N}$ (where k defines the iteration of the Schwarz algorithm) is written:

$$\left\{ \begin{array}{lcll}
\mathcal{L}(W_i^{(k+1)}) & = & f_i & \text{in } \Omega_i \\
C_{n_{ij}} W_i^{(k+1)} & = & C_{n_{ij}} W_j^{(k)} & \text{on } \Gamma_{ij} = \partial\Omega_i \cap \partial\bar{\Omega}_j \quad \text{for} \quad \Omega_i \cap \Omega_j \neq \emptyset \\
A_n^- W_i^{(k+1)} & = & A_n^- g & \text{on } \partial\Omega \cap \partial\Omega_i
\end{array} \right. \tag{4.14}$$

According to the hyperbolic nature of the original system of PDEs and taking into account eq. (4.9), we set $C_{n_{ij}} = A^-_{n_{ij}}$. This choice for the interface operator corresponds to the so-called *classical* or *natural* interface conditions.

Domain decomposition algorithms of the form (4.14) have been extensively studied by Nataf[108], for convection-diffusion problems and Engquist and Zhao[45] for elliptic equations. In particular, these authors have considered the use of high-order optimal interface conditions, inspired from the concept of absorbing boundary conditions for unbounded domains[44], for improving the convergence of the Schwarz algorithm.

For general linear hyperbolic systems of PDEs, whatever the space dimension is, we have the following result due to Clerc[32] who proves the convergence of the Schwarz algorithm in the case of non-overlapping decompositions.

Theorem 2 *Let us denote by $E_i^k = W_i^k - W_{|\partial\Omega_i}$ the error vector associated to the restriction to subdomain Ω_i of the global solution of the problem. Then, the Schwarz algorithm converges in the following sense :*

$$\lim_{k\to\infty} \|E_i^{(k)}\|_{L^2(\Omega_i)^p} = 0 \quad \textit{and} \quad \lim_{k\to\infty} \|\sum_{j=1}^{d} A_j \frac{\partial E_i^{(k)}}{\partial x_j}\|_{L^2(\Omega_i)^p} = 0$$

4.1.3 Convergence analysis for the Euler equations

In this section, we study the convergence of the proposed additive Schwarz algorithm (4.14) based on the classical interface conditions $C_{ij} = A^-_{n_{ij}}$ when applied to the solution of the 2D and 3D Euler equations that model inviscid compressible flows. We consider both overlapping and non-overlapping decompositions. First, we recall the expression of the Euler equations in the two-dimensional case. Then, we consider a two-subdomain decomposition and study the convergence of the Schwarz algorithm in the two- and three-dimensional cases assuming subsonic flow conditions. Let us recall that in the multidimensional supersonic case, the Schwarz algorithm converges in two steps for a two-subdomain decomposition. A Fourier analysis applied to the linearized Euler equations allows us to derive the convergence rate of the ξ-th Fourier component of the error vector. Finally, we conclude this section by studying a three-subdomain decomposition in the two-dimensional case.

The conservative form of the Euler equations in the two-dimensional case is given by:

$$\frac{\partial W}{\partial t} + \vec{\nabla}.\vec{\mathbb{F}}^c(W) = 0 \quad , \quad W = \left(\rho \ , \ \rho\vec{V} \ , \ E\right)^T \quad , \quad \vec{\nabla} = \left(\frac{\partial}{\partial x} \ , \ \frac{\partial}{\partial y}\right)^T \tag{4.15}$$

In eq. (4.15), $W = W(\vec{x}, t)$ where $\vec{x}$ and t respectively denote the spatial and temporal variables. The components of the conservative flux $\vec{\mathbb{F}}^c(W) = (F_1(W), F_2(W))^T$ write as:

$$F_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix} \quad , \quad F_2(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}$$

Note that we have adopted here a notation which is slightly different than the one used in eq. (1.2) of section 1.1 (that is, we use $F_1(W)$ instead of $F_x^c(W)$ and so on). In the above expressions, ρ is the density, $\vec{V} = (u \ , \ v)^T$ is the velocity vector, E is the total energy per unit of volume and p is the pressure. The pressure is deduced from the other variables using the state equation for a perfect gas:

$$p = (\gamma_e - 1)(E - \frac{1}{2}\rho \parallel \vec{V} \parallel^2)$$

where γ_e is the ratio of specific heats ($\gamma_e = 1.4$ for the air).

4.1.3.1 The two-subdomain case

Here, we consider a decomposition of the real space $\mathbb{R}^d$ ($d = 2$ or 3) into two overlapping subdomains such that $\Omega_1 =]-\infty, \gamma[\times\mathbb{R}^{d-1}$ and $\Omega_2 =]\beta, +\infty[\times\mathbb{R}^{d-1}$ where $\beta < \gamma$.

The starting point of our analysis is given by the linearized form (4.3) of the Euler equations:

$$\mathcal{L}(W) \equiv \frac{\mathsf{Id}}{\Delta t} W + A_1 \frac{\partial W}{\partial x_1} + \sum_{i=2}^{d} A_i \frac{\partial W}{\partial x_i} = f \tag{4.16}$$

with $A_i \equiv A_i(\overline{W})$ where $\overline{W}$ denotes the constant vector state used for the linearization of the Euler equations. First, we apply to the above system the change of variable $U = T^{-1}W$ which is based upon the eigenvector factorization of $A_1 = T\Lambda T^{-1}$. Then, (4.16) becomes:

$$\tilde{\mathcal{L}}(U) \equiv bU + B_1 \frac{\partial U}{\partial x_1} + \sum_{i=2}^{d} B_i \frac{\partial U}{\partial x_i} = T^{-1} f \quad , \quad b = \frac{1}{\Delta t} \tag{4.17}$$

where $B_i = T^{-1} A_i T$ and $B_1 = \Lambda = \mathsf{diag}(\lambda_i)$ is the diagonal matrix deduced from the diagonalization of A_1. As will be seen in the following, eq. (4.17) is the symmetrized form of the Euler equations corresponding to eq. (4.5)-(4.6).

In order to estimate the convergence rate of the Schwarz algorithm (4.14), we need to solve local boundary value problems in each subdomain. In the present case where we consider the solution of the two- or three-dimensional Euler equations, we cannot do this directly. The mathematical tool that will allow us to overcome this difficulty is the Fourier transform. More precisely, we now proceed to a Fourier transform (denoted by $\mathcal{F}$) of all the spatial directions except the first one. The vector of Fourier variables is denoted by $\boldsymbol{\xi} = (\xi_j,\, j = 1, \ldots, d-1)$. Let $(E_i^{(k)})(x) = (U_i^{(k)} - U_{|\partial\Omega_i}(x)$ be the error vector in subdomain Ω_i at the k-th iteration of the Schwarz algorithm. We denote by:

$$\hat{E}(x_1, \xi_1, \ldots, \xi_{d-1}) = \mathcal{F}E(x_1, \ldots, x_d) = \int_{\mathbb{R}} e^{-i\xi_1 x_2 - \ldots - i\xi_{d-1} x_d} E(x_1, \ldots, x_d) dx_2 \ldots dx_d$$

the Fourier symbol of the error vector. This transformation can be done only if the A_i matrices are constant which is the case here because we have considered the linearized form of the Euler equations around a constant state $\overline{W}$. The Schwarz algorithm in the Fourier space can be written as follows:

$$\begin{array}{rcl}
\Omega_1 \quad : & \left\{ \begin{array}{rcl} \dfrac{d\hat{E}_1^{(k+1)}}{dx_1} & = & -\mathcal{M}(\boldsymbol{\xi}) \hat{E}_1^{(k+1)} \quad \text{for} \quad x_1 < \gamma \\ (\hat{E}_1^{(k+1)})_j & = & (\hat{E}_2^{(k)})_j \quad \text{for} \quad \lambda_j < 0 \;\; \text{at} \;\; x_1 = \gamma \end{array} \right. \\
\Omega_2 \quad : & \left\{ \begin{array}{rcl} \dfrac{d\hat{E}_2^{(k+1)}}{dx_1} & = & -\mathcal{M}(\boldsymbol{\xi}) \hat{E}_2^{(k+1)} \quad \text{for} \quad x_1 > \beta \\ (\hat{E}_2^{(k+1)})_j & = & (\hat{E}_1^{(k)})_j \quad \text{for} \quad \lambda_j > 0 \;\; \text{at} \;\; x_1 = \beta \end{array} \right.
\end{array} \tag{4.18}$$

where:

$$\mathcal{M}(\boldsymbol{\xi}) = B_1^{-1} (b\mathsf{Id} + i \sum_{m=2}^{d} B_m \xi_{m-1}) \tag{4.19}$$

In (4.18), the subscript j denotes the component of the error vector that must be imposed at a subdomain interface. We obtain local problems that for a given $\boldsymbol{\xi}$ are ODEs whose solutions can be expressed as linear combinations of the eigenvectors of $\mathcal{M}(\boldsymbol{\xi})$:

$$\hat{E}_m^{(k)}(x_1, \boldsymbol{\xi}) = \sum_{j=1}^{p} (\alpha_j^m)^{(k)}(\boldsymbol{\xi}) e^{-\mu_j(\boldsymbol{\xi}) x_1} V_j(\boldsymbol{\xi}) \tag{4.20}$$

where $\mu_j(\boldsymbol{\xi})$ are the eigenvalues of $\mathcal{M}(\boldsymbol{\xi})$. Here we have assumed that the eigenvectors $V_j(\boldsymbol{\xi})$ of $\mathcal{M}(\boldsymbol{\xi})$ are linearly independent. Furthermore, we require that these local solutions are bounded at infinity ($-\infty$ and $+\infty$ respectively)

which implies that, in the decomposition of $\hat{E}_1(x_1, \boldsymbol{\xi})$ (respectively $\hat{E}_2(x_1, \boldsymbol{\xi})$), we need to use the eigenvectors corresponding to the negative (respectively the positive) eigenvalues. Then, we replace the expressions of the local solutions (4.20) into the interface conditions (4.18) which results in:

$$\left(\sum_{j,\Re(\mu_j)<0} (\alpha_j^1(\boldsymbol{\xi}))^{(k+1)} e^{-\mu_j(\boldsymbol{\xi})\gamma} V_j(\boldsymbol{\xi}) \right)_l = \left(\sum_{j,\Re(\mu_j)>0} (\alpha_j^2(\boldsymbol{\xi}))^{(k)} e^{-\mu_j(\boldsymbol{\xi})\gamma} V_j(\boldsymbol{\xi}) \right)_l , \ \Re(\lambda_l) < 0 \quad \text{In } \Omega_1$$

$$\left(\sum_{j,\Re(\mu_j)>0} (\alpha_j^2(\boldsymbol{\xi}))^{(k+1)} e^{-\mu_j(\boldsymbol{\xi})\beta} V_j(\boldsymbol{\xi}) \right)_l = \left(\sum_{j,\Re(\mu_j)<0} (\alpha_j^1(\boldsymbol{\xi}))^{(k)} e^{-\mu_j(\boldsymbol{\xi})\beta} V_j(\boldsymbol{\xi}) \right)_l , \ \Re(\lambda_l) > 0 \quad \text{In } \Omega_2 \tag{4.21}$$

By solving the above equations for the coefficients α_j^m, we obtain the following general form of interface iterations:

$$\begin{cases} (\alpha_j^1)_{j,\Re(\mu_j)<0}^{(k+1)}(\boldsymbol{\xi}) &= \mathcal{T}_1 (\alpha_j^2)_{j,\Re(\mu_j)>0}^{(k)}(\boldsymbol{\xi}) \\ (\alpha_j^2)_{j,\Re(\mu_j)>0}^{(k+1)}(\boldsymbol{\xi}) &= \mathcal{T}_2 (\alpha_j^1)_{j,\Re(\mu_j)<0}^{(k)}(\boldsymbol{\xi}) \end{cases} \tag{4.22}$$

Then, the square of the convergence rate of the $\boldsymbol{\xi}$-th component of the error vector of the Schwarz algorithm can be computed as the spectral radius of one of the matrix products $\mathcal{T}_1\mathcal{T}_2(\boldsymbol{\xi})$ or $\mathcal{T}_2\mathcal{T}_1(\boldsymbol{\xi})$:

$$\rho_2^2 \equiv \rho_{\mathsf{Schwarz2}}^2 = \rho(\mathcal{T}_1\mathcal{T}_2) = \rho(\mathcal{T}_2\mathcal{T}_1) \tag{4.23}$$

4.1.3.2 The two-dimensional case

As a first step, we apply the general methodology described previously to the solution of two-dimensional Euler equations. In that case $\boldsymbol{\xi} \equiv \xi$ is a scalar and the linearized form (4.17) is characterized by:

$$B_1 = \mathsf{diag}(u-c \ , \ u+c \ , \ u \ , \ u) \qquad B_2 = \begin{pmatrix} v & 0 & \frac{c}{\sqrt{2}} & 0 \\ 0 & v & \frac{c}{\sqrt{2}} & 0 \\ \frac{c}{\sqrt{2}} & \frac{c}{\sqrt{2}} & v & 0 \\ 0 & 0 & 0 & v \end{pmatrix} \tag{4.24}$$

where we have assumed that $W = \left(\rho \ , \ \rho\vec{V} \ , \ E\right)^T$ denotes the constant vector state used for the linearization of the Euler equations. The matrix $\mathcal{M}(\xi)$ corresponding to (4.19) is written:

$$\mathcal{M}(\xi) = \begin{pmatrix} \frac{a}{u-c} & 0 & \frac{i\xi c}{\sqrt{2}(u-c)} & 0 \\ 0 & \frac{a}{u+c} & \frac{i\xi c}{\sqrt{2}(u+c)} & 0 \\ \frac{i\xi c}{\sqrt{2}u} & \frac{i\xi c}{\sqrt{2}u} & \frac{a}{u} & 0 \\ 0 & 0 & 0 & \frac{a}{u} \end{pmatrix} \tag{4.25}$$

with $a = b + i\xi v$.

We obtain the following expressions for the eigenvalues and the corresponding eigenvectors of the matrix $\mathcal{M}(\xi)$:

$$\left\{\begin{array}{l} \mu_1(\xi) = \dfrac{-au - cR(\xi)}{c^2-u^2} \quad , \quad V_1(\xi) = \left[-\dfrac{(R(\xi)+a)(c+u)}{\sqrt{2}}, \dfrac{(R(\xi)-a)(c-u)}{\sqrt{2}}, i\xi(c^2-u^2), 0\right]^T \\ \mu_2(\xi) = \dfrac{-au + cR(\xi)}{c^2-u^2} \quad , \quad V_2(\xi) = \left[\dfrac{(R(\xi)-a)(c+u)}{\sqrt{2}}, -\dfrac{(R(\xi)+a)(c-u)}{\sqrt{2}}, i\xi(c^2-u^2), 0\right]^T \\ \mu_{3,4}(\xi) = \dfrac{a}{u} \quad , \quad V_3(\xi) = \left[-\dfrac{i\xi u}{\sqrt{2}}, \dfrac{i\xi u}{\sqrt{2}}, a, 0\right]^T \quad , \quad V_4(\xi) = [0,0,0,1]^T \end{array}\right.$$

where $R(\xi) = \sqrt{a^2 + \xi^2(c^2-u^2)}$.

Remark 1 *A very simple calculation shows that the eigenvectors are linearly dependent only for $v = 0$ and $\xi^2 = \dfrac{b^2}{u^2}$ (in that case, two of the eigenvalues are equal and the matrix $\mathcal{M}(\xi)$ cannot be diagonalized by eigenvectors). In order to get a more general result, we should apply rigorously the solution proposed in [44] but since the eigenvectors are linearly dependent in a very specific case, we adopt the method previously described and assume in the sequel that $\xi^2 \neq \dfrac{b^2}{u^2}$.*

At that point, we make the assumption that the flow is subsonic that is, the local Mach number defined by $M = \dfrac{\sqrt{u^2+v^2}}{c}$ is such that $M < 1$. Note that this also means that $\dfrac{|u|}{c} < 1$ and $\dfrac{|v|}{c} < 1$. Finally, we also assume that the flow is such that $u > 0$ and thus, we have that $0 < u < c$. Under these conditions, one can verify that $\Re(\mu_1) < 0$ and $\Re(\mu_{2,3,4}) > 0$. For the sake of a better interpretation of the results of the analysis, we introduce the dimensionless wave number defined by $\bar{\xi} = \dfrac{c\xi}{b}$ and the associated dimensionless quantities:

$$\left\{\begin{array}{lcl} \bar{a} & = & \dfrac{b}{c} + i\bar{\xi}M_t \; = \; \dfrac{a}{c} \\ \bar{R}(\xi) & = & \sqrt{\bar{a}^2 + \bar{\xi}^2(1-M_n^2)} \; = \; \dfrac{R(\bar{\xi})}{c} \\ \mu_{1,2} & = & \dfrac{c}{b}(\mu_1 - \mu_2) = -\dfrac{2\bar{R}(\bar{\xi})}{1-M_n^2} \\ \mu_{1,3} & = & \dfrac{c}{b}(\mu_1 - \mu_3) = -\dfrac{\bar{a} + M_n\bar{R}(\bar{\xi})}{M_n(1-M_n^2)} \end{array}\right.$$

where $M_n = \dfrac{u}{c}$ and $M_t = \dfrac{v}{c}$ respectively denote the local normal and tangential Mach numbers. Then, by solving (4.21) and using (4.23) we obtain the square of the convergence rate of the algorithm:

$$\rho_2^2(\bar{\xi},\bar{\delta}) = \left| \frac{[\bar{R}(\bar{\xi}) - \bar{a}]^2[\bar{a} + M_n\bar{R}(\bar{\xi})]}{[\bar{R}(\bar{\xi}) + \bar{a}]^2[\bar{a} - M_n\bar{R}(\bar{\xi})]} e^{\mu_{1,2}\bar{\delta}} - \frac{2[\bar{R}(\bar{\xi}) - \bar{a}][M_n(1-M_n)]\bar{R}(\bar{\xi})}{[\bar{R}(\bar{\xi}) + \bar{a}][1+M_n][\bar{a}c - M_n\bar{R}(\bar{\xi})]} e^{\mu_{1,3}\bar{\delta}} \right| \tag{4.26}$$

where:

$$\bar{\delta} = \frac{\gamma - \beta}{c\Delta t}$$

is the dimensionless size of the overlapping area.

The following Proposition concerns the convergence of the additive Schwarz algorithm (4.14) applied to the solution of the two-dimensional Euler equations (1.1) that is the inequality $\rho_2^2(\bar{\xi},\bar{\delta}) < 1$.

Proposition 1 *In the non-overlapping case, $\bar{\delta} = 0$, we have $\rho_2^2(\bar{\xi},0) < 1$ for all $\bar{\xi}$, M_n and M_t. In the overlapping case, there exists $\bar{\delta}_0 > 0$ such that $\rho_2^2(\bar{\xi},\bar{\delta}) < 1$ for all $\bar{\delta} > \bar{\delta}_0$, $\bar{\xi}$, M_n and M_t.*

The proof is given in [42].

We recall that the above result has already been proved only in the non-overlapping case by Clerc[32] using an energy estimate method.

For small values of the overlap, we needed additional assumptions on the velocity field. Numerical experiments indicate that these assumptions are not necessary. The difficulty in proving a general convergence result comes from the fact that the convergence rate is not decreasing with respect to the size of the overlap. For some small values of $\bar{\delta}$, we can find at least one pair of $(M_n\ ,\ M_t)$ and an interval of wave numbers $[\bar{\xi}_1,\bar{\xi}_2]$ such that :

$$\rho_2^2(\bar{\xi},\bar{\delta}) > \rho_2^2(\bar{\xi},0) \quad \text{for} \quad \bar{\xi} \in [\bar{\xi}_1,\bar{\xi}_2],$$

as can be seen on figure 4.1 which illustrates this behavior for $(M_n = 0.3\ ,\ M_t = 0.01)$. This behavior is very different from the one characterizing the scalar case where the convergence rate is a decreasing function of the size of the overlap, see for example [77].

The convergence rate does not depend on μ_4 since, as one can see from (4.25), the fourth component of $U = T^{-1}W$ is decoupled from the others.

With the discrete Courant number :

$$\mathsf{CFL}_h = \frac{(M+1)c}{bh} = \frac{(M+1)c\Delta t}{h} \tag{4.27}$$

where h denotes a characteristic dimension of the grid used for space discretization, we see that if the overlap between subdomains is equal to h, then the dimensionless overlap size $\bar{\delta}$ found in the expression of the convergence rate is of the order $\dfrac{1}{\mathsf{CFL}_h}$. Figure 4.1 shows that for large discrete Courant number, the overlap may decrease the performance of the algorithm.

For the non-overlapping case, Figure 4.2 represents the norm $\|\rho_2\|_\infty$ for a given value of the tangential Mach number, as a function of the normal Mach numbers M_n, and shows the global behavior of the convergence rate for different pairs of $(M_n\ ,\ M_t)$. In addition, the convergence rate as $\xi \to \infty$ satisfies :

$$\lim_{\xi\to+\infty} \rho_2(\xi) = \sqrt{\left(\frac{1-3M_n}{1+M_n}\right)^2 + \frac{8M_nM_t^2}{(1+M_n)^3}} \quad < \quad 1 \tag{4.28}$$

In the particular case where $M_n = \dfrac{1}{3}$ and $M_t = 0$, this limit becomes null. This is surprising and certainly not expected. It is obtained for $v = 0$ everywhere in the flow field which is very particular and probably ideal situation.

We note that we have limited our analysis to a subsonic flow which is the case of interest from the point of view of information propagation at subdomain interfaces.

Remark 2 *The inequality (4.28) has a numerical meaning: for a given discretization, let $k_{\max}$ denote the largest frequency that can be represented on a grid. This largest frequency is of the order $\dfrac{\pi}{h}$ where h denotes a characteristic grid size. The convergence rate of the additive Schwarz algorithm on this grid can be estimated by $\rho_2^h = \max\limits_{|k|<k_{\max}} \rho_2(k)$. From (4.28), we have that $\rho_2^h \leq \max\limits_{k\in\mathbb{R}} \rho_2(k) < 1$ meaning that for finer grids, the number of iterations may increase slightly but should not go to infinity.*

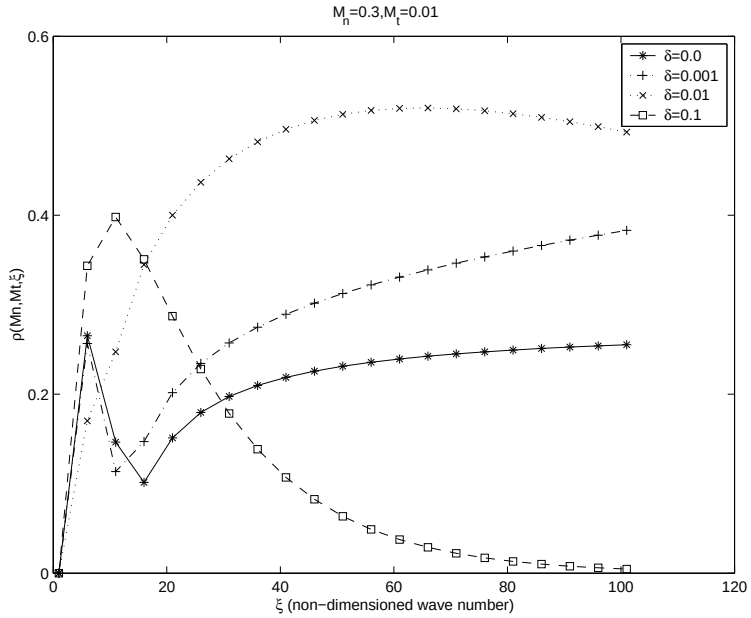


Figure 4.1: Convergence rate of the additive Schwarz algorithm (2D Euler equations)
Two-subdomain decomposition (δ denotes here the dimensionless size of the overlapping area)

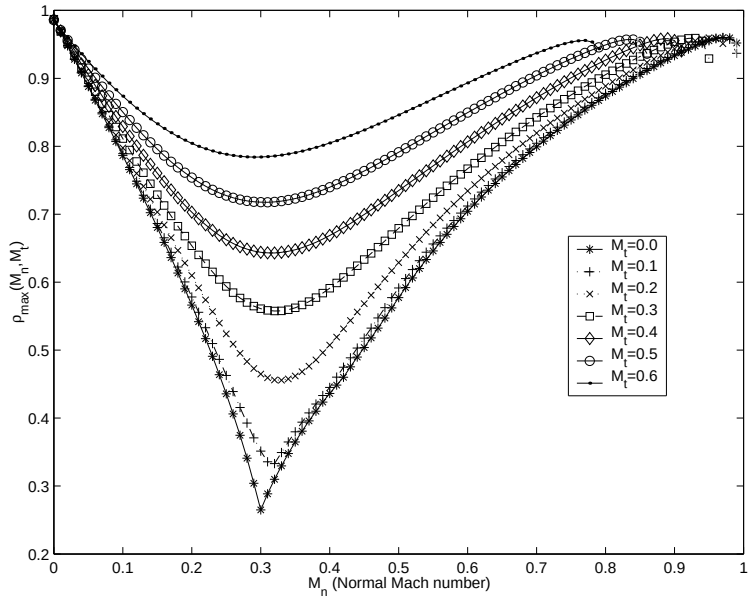


Figure 4.2: Norm $\|\rho_2\|_\infty$ for the additive Schwarz algorithm (2D Euler equations)
Two-subdomain non-overlapping decomposition ($\bar{\delta} = 0$)

The behavior of the convergence rate with respect to the dimensionless wave number $\bar{\xi}$ is represented on figure 4.3 in the case of a non-overlapping decomposition for different values of M_n and M_t. Note that we have only considered situations such that $M_n^2+M_t^2<1$. We remark that the behavior of the Schwarz algorithm deteriorates and becomes less sensitive to the value of the normal Mach number when the tangential Mach number tends to 1.

Remark 3 *For a larger number of subdomains we cannot calculate easily the convergence rate using the previous technique because this leads to evaluate the spectral radius of a $4(N-1)\times 4(N-1)$ matrix where N is the number of subdomains. We will only do this in the three-subdomain case (see section 4.1.3.4).*

4.1.3.3 The three-dimensional case

As in the two-dimensional case, the starting-point of the convergence analysis is given by the linearized system:

$$\tilde{\mathcal{L}}U \equiv bU + B_1\frac{\partial U}{\partial x_1} + B_2\frac{\partial U}{\partial x_2} + B_3\frac{\partial U}{\partial x_3} = T^{-1}f \tag{4.29}$$

where $B_1 = \mathrm{diag}(u-c\ ,\ u+c\ ,\ u\ ,\ u\ ,\ u)$:

$$B_2 = \begin{pmatrix} v & 0 & \frac{c}{\sqrt{2}} & 0 & 0 \\ 0 & v & \frac{c}{\sqrt{2}} & 0 & 0 \\ \frac{c}{\sqrt{2}} & \frac{c}{\sqrt{2}} & v & 0 & 0 \\ 0 & 0 & 0 & v & 0 \\ 0 & 0 & 0 & 0 & v \end{pmatrix} \quad B_3 = \begin{pmatrix} w & 0 & 0 & \frac{c}{\sqrt{2}} & 0 \\ 0 & w & 0 & \frac{c}{\sqrt{2}} & 0 \\ 0 & 0 & w & 0 & 0 \\ \frac{c}{\sqrt{2}} & \frac{c}{\sqrt{2}} & 0 & w & c \\ 0 & 0 & 0 & 0 & w \end{pmatrix}$$

are the associated transformed Jacobian matrices.

The matrix $\mathcal{M}(\boldsymbol{\xi})$ corresponding to (4.19) is written:

$$\mathcal{M}(\boldsymbol{\xi}) = \begin{pmatrix} \frac{a(\boldsymbol{\xi})}{u-c} & 0 & \frac{1}{\sqrt{2}}\frac{i\xi_1 c}{u-c} & \frac{1}{\sqrt{2}}\frac{i\xi_2 c}{u-c} & 0 \\ 0 & \frac{a(\boldsymbol{\xi})}{u+c} & \frac{1}{\sqrt{2}}\frac{i\xi_1 c}{u+c} & \frac{1}{\sqrt{2}}\frac{i\xi_2 c}{u+c} & 0 \\ \frac{1}{\sqrt{2}}\frac{i\xi_1 c}{u} & \frac{1}{\sqrt{2}}\frac{i\xi_1 c}{u} & \frac{a(\boldsymbol{\xi})}{u} & 0 & 0 \\ \frac{1}{\sqrt{2}}\frac{i\xi_2 c}{u} & \frac{1}{\sqrt{2}}\frac{i\xi_2 c}{u} & 0 & \frac{a(\boldsymbol{\xi})}{u} & 0 \\ 0 & 0 & 0 & 0 & \frac{a(\boldsymbol{\xi})}{u} \end{pmatrix} \tag{4.30}$$

with $a(\boldsymbol{\xi}) = b + i\xi_1 v + i\xi_2 w$ and $\boldsymbol{\xi} = (\xi_1, \xi_2)$. We obtain the following expressions for the eigenvalues and associated eigenvectors:

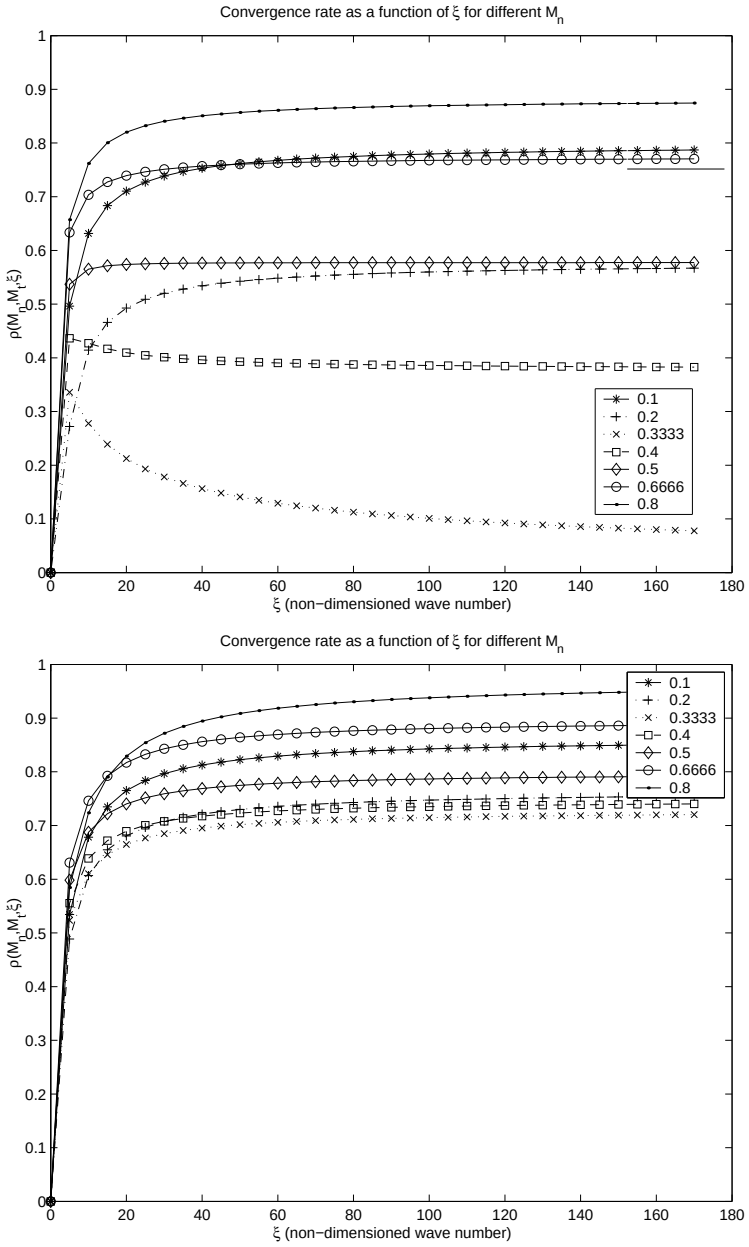


Figure 4.3: Convergence rate of the additive Schwarz algorithm (2D Euler equations)
Two-subdomain non-overlapping decomposition ($\bar{\delta} = 0$)
Top : $M_t = 0$ - Bottom : $M_t = 0.5$

$$\left\{\begin{array}{lcl}
\mu_1(\boldsymbol{\xi}) & = & \dfrac{-au - cR(\boldsymbol{\xi})}{c^2-u^2} \\
V_1(\boldsymbol{\xi}) & = & \left[-\dfrac{(R(\boldsymbol{\xi})+a)(c+u)}{\sqrt{2}}, \dfrac{(R(\boldsymbol{\xi})-a)(c-u)}{\sqrt{2}}, i\xi_1(c^2-u^2), i\xi_2(c^2-u^2), 0\right]^T \\
\mu_2(\boldsymbol{\xi}) & = & \dfrac{-au + cR(\boldsymbol{\xi})}{c^2-u^2} \\
V_2(\boldsymbol{\xi}) & = & \left[\dfrac{(R(\boldsymbol{\xi})-a)(c+u)}{\sqrt{2}}, -\dfrac{(R(\boldsymbol{\xi})+a)(c-u)}{\sqrt{2}}, i\xi_1(c^2-u^2), i\xi_2(c^2-u^2), 0\right]^T \\
\mu_{3,4,5}(\boldsymbol{\xi}) & = & \dfrac{a}{u} \\
V_3(\boldsymbol{\xi}) & = & \left[-\dfrac{i\xi_1 u}{\sqrt{2}}, \dfrac{i\xi_1 u}{\sqrt{2}}, a, 0, 0\right]^T \\
V_4(\boldsymbol{\xi}) & = & [0, 0, -\xi_2, \xi_1, 0]^T \\
V_5(\boldsymbol{\xi}) & = & [0, 0, 0, 0, 1]^T
\end{array}\right. \tag{4.31}$$

where $R(\boldsymbol{\xi}) = \sqrt{a^2 + (\xi_1^2+\xi_2^2)(c^2-u^2)}$. We note that we can derive the expression of the convergence rate in the 3D case in the same way as we did for the two-dimensional case and we get:

$$\rho_2^2(\bar{\boldsymbol{\xi}}, \bar{\delta}) = \left| \frac{[\bar{R}(\bar{\boldsymbol{\xi}}) - \bar{a}]^2[\bar{a} + M_n\bar{R}(\bar{\boldsymbol{\xi}})]}{[\bar{R}(\bar{\boldsymbol{\xi}}) + \bar{a}]^2[\bar{a} - M_n\bar{R}(\bar{\boldsymbol{\xi}})]} e^{\mu_{1,2}\bar{\delta}} - \frac{2[\bar{R}(\bar{\boldsymbol{\xi}}) - \bar{a}][M_n(1-M_n)]\bar{R}(\bar{\boldsymbol{\xi}})}{[\bar{R}(\bar{\boldsymbol{\xi}}) + \bar{a}][1+M_n][\bar{a}c - M_n\bar{R}(\bar{\boldsymbol{\xi}})]} e^{\mu_{1,3}\bar{\delta}} \right| \tag{4.32}$$

where $\bar{\boldsymbol{\xi}} = (\bar{\xi_1}, \bar{\xi_2}) = \left(\dfrac{c\xi_1}{b}, \dfrac{c\xi_2}{b}\right)$ is the vector of dimensionless wave numbers and:

$$\bar{a}(\bar{\boldsymbol{\xi}}) = 1 + i\bar{\xi_1}M_{t1} + i\bar{\xi_2}M_{t2} \quad , \quad M_{t1} = \frac{v}{c} \quad , \quad M_{t2} = \frac{w}{c}$$

where M_{t1} and M_{t2} denote the tangential local Mach number on the interface. We note that this result could have been obtained simply by replacing in (4.26) $\bar{a}(\bar{\xi})$ with $\bar{a}(\bar{\boldsymbol{\xi}})$ and $\bar{R}(\bar{\xi})$ with $\bar{R}(\bar{\boldsymbol{\xi}})$. As a consequence, the asymptotic behavior of the convergence rate is independent of the space dimension.

4.1.3.4 The three-subdomain case in the two-dimensional case

We now consider the case of a decomposition in three overlapping or non-overlapping subdomains in the context of the solution of the two-dimensional Euler equations : $\Omega_1 =]-\infty\ ,\ \beta[\times\mathbb{R}$, $\Omega_2 =]\beta'\ ,\ \gamma[\times\mathbb{R}$ and $\Omega_3 =]\gamma'\ ,\ +\infty[\times\mathbb{R}$ where $\beta' \leq \beta$ and $\gamma' \leq \gamma$. Proceeding in a similar way as previously, we estimate the convergence rate of the additive Schwarz algorithm (4.14) by means of a Fourier transform technique.

In terms of the local error vectors, the Schwarz algorithm is written:

$$
\begin{aligned}
\Omega_1 \quad : \quad & \left\{ \begin{array}{rcl} \dfrac{d\hat{E}_1^{(k+1)}}{dx_1} & = & -\mathcal{M}(\boldsymbol{\xi})\hat{E}_1^{(k+1)} \quad \text{for} \quad x_1 < \beta \\ (\hat{E}_1^{(k+1)})_j & = & (\hat{E}_2^{(k)})_j \quad \text{for} \quad \lambda_j < 0 \ \text{ and } \ x_1 = \beta \end{array} \right. \\
\Omega_2 \quad : \quad & \left\{ \begin{array}{rcl} \dfrac{d\hat{E}_2^{(k+1)}}{dx_1} & = & -\mathcal{M}(\boldsymbol{\xi})\hat{E}_2^{(k+1)} \quad \text{for} \quad \beta < x_1 < \gamma \\ (\hat{E}_2^{(k+1)})_j & = & (\hat{E}_1^{(k)})_j \quad \text{for} \quad \lambda_j > 0 \ \text{ and } \ x_1 = \beta' \\ (\hat{E}_2^{(k+1)})_j & = & (\hat{E}_3^{(k)})_j \quad \text{for} \quad \lambda_j < 0 \ \text{ and } \ x_1 = \gamma \end{array} \right. \\
\Omega_3 \quad : \quad & \left\{ \begin{array}{rcl} \dfrac{d\hat{E}_3^{(k+1)}}{dx_1} & = & -\mathcal{M}(\boldsymbol{\xi})\hat{E}_3^{(k+1)} \quad \text{for} \quad x_1 > \gamma' \\ (\hat{E}_3^{(k+1)})_j & = & (\hat{E}_2^{(k)})_j \quad \text{for} \quad \lambda_j > 0 \ \text{ and } \ x_1 = \gamma' \end{array} \right.
\end{aligned}
\tag{4.33}
$$

Similarly to the two-subdomain case, we take into account the behavior of the local solutions at $\pm\infty$ (i.e. $\hat{E}_1^{(k+1)}$ and $\hat{E}_3^{(k+1)}$ are bounded respectively at $-\infty$ and $+\infty$). We make the same assumptions concerning the nature of the flow that is, $M < 1$ and $0 < u < c$. Then, we obtain the local solutions as combinations of the eigenvectors of the matrix $\mathcal{M}(\xi)$ (see equation (4.25)):

$$
\left\{ \begin{array}{rcl}
\hat{E}_1^{(k)}(x,\xi) & = & (\alpha_1^1)^{(k)} e^{-\mu_1(\xi)x} V_1(\xi) \\
\hat{E}_2^{(k)}(x,\xi) & = & (\alpha_1^2)^{(k)} e^{-\mu_1(\xi)x} V_1(\xi) + (\alpha_2^2)^{(k)} e^{-\mu_2(\xi)x} V_2(\xi) \\
& + & (\alpha_3^2)^{(k)} e^{-\mu_3(\xi)x} V_3(\xi) + (\alpha_4^2)^{(k)} e^{-\mu_4(\xi)x} V_4(\xi) \\
\hat{E}_3^{(k)}(x,\xi) & = & (\alpha_2^3)^{(k)} e^{-\mu_2(\xi)x} V_2(\xi) + (\alpha_3^3)^{(k)} e^{-\mu_3(\xi)x} V_3(\xi) + (\alpha_4^3)^{(k)} e^{-\mu_4(\xi)x} V_4(\xi)
\end{array} \right.
\tag{4.34}
$$

where the upper index in α_j^i identifies the subdomain number. We note that if we know the $(\alpha_j^2)^{(k)}$ for $j = 1, 2, 3, 4$ then we can determine $(\alpha_1^1)^{(k+1)}$ and $(\alpha_j^3)^{(k+1)}$ for $j = 2, 3, 4$ by solving the local problems in Ω_1 and Ω_3. Reversely, if we have $(\alpha_1^1)^{(k)}$ and $(\alpha_j^3)^{(k)}$ for $j = 2, 3, 4$ by solving the local problem in subdomain Ω_2 we get $(\alpha_j^2)^{(k+1)}$ for $j = 1, 2, 3, 4$. Therefore by solving the problems defined in each subdomain we obtain the expression of the interface iterations in terms of the α_j^m:

$$
\left\{ \begin{array}{rcl}
\begin{bmatrix} \alpha_1^1 \\ \alpha_2^3 \\ \alpha_3^3 \\ \alpha_4^3 \end{bmatrix}^{(k+1)} & = & \mathcal{T}_1 \begin{bmatrix} \alpha_1^2 \\ \alpha_2^2 \\ \alpha_3^2 \\ \alpha_4^2 \end{bmatrix}^{(k)} = \mathcal{T}_1\mathcal{T}_2 \begin{bmatrix} \alpha_1^1 \\ \alpha_2^3 \\ \alpha_3^3 \\ \alpha_4^3 \end{bmatrix}^{(k-1)} \\
\begin{bmatrix} \alpha_1^2 \\ \alpha_2^2 \\ \alpha_3^2 \\ \alpha_4^2 \end{bmatrix}^{(k+1)} & = & \mathcal{T}_2 \begin{bmatrix} \alpha_1^1 \\ \alpha_2^3 \\ \alpha_3^3 \\ \alpha_4^3 \end{bmatrix}^{(k)} = \mathcal{T}_2\mathcal{T}_1 \begin{bmatrix} \alpha_1^2 \\ \alpha_2^2 \\ \alpha_3^2 \\ \alpha_4^2 \end{bmatrix}^{(k-1)}
\end{array} \right.
\tag{4.35}
$$

where the $\mathcal{T}_{1,2}$ are matrices with complex entries. As in the two-dimensional case, the square of the convergence rate of the algorithm $\rho_3^2 \equiv \rho^2_{\text{Schwarz3}}$ is given by the spectral radius of one of the matrices $\mathcal{T}_1\mathcal{T}_2$ or $\mathcal{T}_2\mathcal{T}_1$.

After some algebraic calculations and by introducing the dimensionless wave number $\bar{\xi} = \dfrac{c\xi}{b}$ we obtain the following expression for the convergence rate:

$$\rho_3^2(\bar{\xi}, \bar{\delta}_1, \bar{\delta}_2, \bar{L}) = \left| \frac{\rho_2^2(\bar{\xi}, \bar{\delta}_1) + \rho_2^2(\bar{\xi}, \bar{\delta}_2) - 2\rho_2^2(\bar{\xi}, \bar{L}) + \sqrt{F}}{2(1 - \rho_2^2(\bar{\xi}, \bar{L}))} \right| \tag{4.36}$$

where $\bar{\delta}_1 = \dfrac{\beta - \beta'}{c\Delta t}$ and $\bar{\delta}_2 = \dfrac{\gamma - \gamma'}{c\Delta t}$ denote the dimensionless size of the overlapping zones between the subdomains and $\bar{L} = \dfrac{\gamma - \beta'}{c\Delta t}$ is the dimensionless width of the second subdomain. In the above expression, $\rho_2(\bar{\xi}, \bar{\delta})$ is the convergence rate of the Schwarz algorithm in the two-subdomain case (see equation (4.26)) with an overlapping zone of size $\bar{\delta}$ and:

$$F = \sqrt{(\rho_2^2(\bar{\xi}, \delta_1) + \rho_2^2(\bar{\xi}, \delta_2) - 2\rho_2^2(\bar{\xi}, \bar{L}))^2 - 4(1 - \rho_2^2(\bar{\xi}, \bar{L}))(\rho_2^2(\bar{\xi}, \delta_1)\rho_2^2(\bar{\xi}, \delta_2) - \rho_2^2(\bar{\xi}, \bar{L}))}$$

We can further assume that the overlap is the same for each pair of subdomains $\bar{L}_1 = \bar{L}_2 = \bar{\delta}$. With this hypothesis the expression of the convergence rate becomes:

$$\rho_3^2(\bar{\xi}, \bar{\delta}, \bar{L}) = \left| \frac{\rho_2^2(\bar{\xi}, \bar{\delta}) + \rho_2(\bar{\xi}, \bar{L})}{1 + \rho_2(\bar{\xi}, \bar{L})} \right| \tag{4.37}$$

Proposition 2 *There exists $\bar{L}_0 > 0$ such that for $\bar{L} > \bar{L}_0$ we have $\rho_3^2(\bar{\xi}, \bar{\delta}, \bar{L}) < 1$ under the same assumptions on the size of the overlap and the velocity field as in Proposition 1.*

The proof is given in [42].

In the non-overlapping case ($\bar{\delta} = 0$) we have the following properties:

- the asymptotic value as $\bar{\xi}$ tends to infinity is the same as in the two-subdomain case that is:

$$\lim_{\bar{\xi} \to +\infty} \rho_3^2(\bar{\xi}, 0, \bar{L}) = \lim_{\bar{\xi} \to +\infty} \rho_2^2(\bar{\xi}, 0)$$

- the following inequality (which holds for $\bar{L}$ sufficiently large):

$$\rho_3^2(\bar{\xi}, 0, \bar{L}) > \rho_2^2(\bar{\xi}, 0)$$

  shows that the convergence is slower when the number of subdomains increases and, as the size of the middle subdomain becomes small, we have:

$$\lim_{\bar{L} \to 0} \rho_3^2(\bar{\xi}, 0, 0, \bar{L}) = \rho_2(\bar{\xi}, 0)$$

4.1.3.5 Summary

From the results of the analyses undertaken in the previous sections we can conclude that the additive Schwarz algorithm (4.14) demonstrates a qualitatively similar behavior, irrespectively of the number of subdomains, when dealing with high frequencies and non-overlapping decompositions. This stems from the fact that the expression of the convergence rate in the three-subdomain case can be related to that obtained in the two-subdomain case using an overlapping decomposition. At the same time, these results are independent of the dimension of the problem. It is important to notice that, even in the non-overlapping case, the use of classical transmission conditions is sufficient to obtain a convergent algorithm which was not the case when dealing with scalar problems, see for example [76].

We note that the results obtained in this preliminary convergence analysis have motivated an ongoing study where we apply the Smith factorization theory[60] to the convergence analysis of the proposed algorithm in the two-dimensional case and for a two-subdomain non-overlapping decomposition. This mathematical tool allows us to obtain a formulation of the Schwarz algorithm which is more representative of the system of PDEs under consideration. This intrinsic formulation of the Schwarz algorithm has two implications on our study : on one hand, it allows us to explain the good convergence properties of the Schwarz algorithm based on classical interface conditions and, on the other hand, it is used as the basis for the construction of optimized interface conditions that improve the convergence of the Schwarz algorithm.

4.1.3.6 Numerical assessment of the convergence of the Schwarz algorithm

In this section, we present numerical results concerning the convergence of the additive Schwarz algorithm (4.14) based on the classical interface conditions characterized by transmission operators of the form $C_{ij} = A^-_{n_{ij}}$. This experimental assessment is done in the context of the calculation of compressible flows that are modeled by the two-dimensional Euler equations. Moreover, we only consider the non-overlapping variant of algorithm (4.14). The implementation of the proposed domain decomposition method is described in details in section 4.1.4. We simply recall here the main characteristics of this implementation that are relevant for the interpretation of the results of the numerical experiments considered in the following.

4.1.3.6.1 Brief overview of the implementation of the Schwarz algorithm. The Euler equations are discretized in space by using the mixed element/volume formulation on unstructured triangular meshes which is described in section 1.2.1 of chapter 1. Time integration of the resulting semi-discrete equations makes use of the linearized implicit scheme described in section 1.4.1. Then, at each pseudo-time step, a linear system must be solved to advance the solution in time. This is where the additive Schwarz algorithm (4.14) is introduced. The parallelization strategy adopted in the original flow solver combines a partitioning of the underlying triangular mesh and a message-passing programming model. According to the form of interface conditions used in (4.14), it is interesting to consider mesh partitions involving a one-triangle wide overlapping region that is shared by neighboring subdomains. As a matter of fact, it is easily seen that within this setting, the interface between two neighboring subdomains is a non-overlapping one from the point of view of the dual discretization of Ω in terms of control surfaces (see figure 4.10 of section 4.1.4). Then, if Ω_1 and Ω_2 are neighboring subdomains:

$$\Gamma_{12} = \Omega_1 \cap \Omega_2 = \bigcup_{C_i \in \Omega_1, C_j \in \Omega_2} \partial C_i \cap \partial C_j$$

Remark 4 *Since an approximate Riemann solver such as the one proposed by Roe[120] provides an approximation of an elementary numerical flux at the boundary $\partial C_i \cap \partial C_j$, the above choice for the definition of Γ_{12} allows for a natural introduction of interface unknowns in terms of numerical fluxes as discussed in section 4.1.4. However, the main problem with this choice is that the interface is not a straight line but rather a broken line contrary to the assumption done for the convergence analysis (see section 4.1.3).*

One particularity of our implementation is that we make use of a specific set of interface unknowns that are expressed in terms of positive and negative parts of elementary numerical fluxes computed using the approximate Riemann solver of Roe[120]. In order to do so, we need to consider a preliminary step which consists in the introduction of a redundant variable $W^\star$ at the interface between two control surfaces (see figure 4.10). Clearly, the interface conditions that need to be taken into account in the additive Schwarz algorithm (4.14) are of the form $A^-_{n_{ij}} W^\star$ for Ω_1 and $A^+_{n_{ij}} W^\star$ for Ω_2 (if we assume that the normal vector n_{ij} is directed from Ω_1 to Ω_2). In section 4.1.4 we explain how these interface conditions can be used to modify the linearization of the approximate Riemann solver of Roe[120] for mesh edges $[s_i, s_j]$ such that C_i (associated with s_i) and C_j (associated with s_j) belong to two neighboring subdomains. This approach results in a linearized implicit time integration scheme involving a global Jacobian matrix that has a block structure. As classically done in this context, a sub-structuring technique is applied to this matrix which results in the formulation of a reduced system involving interface unknowns that, in the present case, are expressed in terms of as positive and negative parts of elementary numerical fluxes.

It is shown in [40] that the additive Schwarz algorithm (4.14) is equivalent to a Richardson iterative method acting on the reduced interface system. In practice the solution of this interface system is accelerated using a Krylov method such as GMRES which is more efficient and robust. However, our goal here is to perform numerical experiments that are representative of the conditions of the convergence analysis of section 4.1.3. Therefore, the numerical results presented below are based on the standard approach resulting in the application of a Richardson method to solve the interface system. As usual, each iteration of the Richardson method requires solving independent linear systems in each subdomain. In the present study, a point-wise Gauss-Seidel relaxation method is used for the solution of these local systems. In the following, we will refer to ε_l for the linear threshold associated with the solution of the local systems and to ε_i for the linear threshold associated with the solution of the interface system.

4.1.3.6.2 Test cases definition. The numerical simulations considered here aim at assessing the convergence results of section 4.1.3 from an experimental viewpoint. We thus focus on the solution of the linear system resulting from the first implicit time step starting from a uniform flow. The CFL number is set to 1000 for all the numerical simulations. We consider two geometries: a rectangular domain of size $[0, 8] \times [0, 1]$ and a `NACA0012` airfoil. For the first geometry, two types of discretization are used : the first type consists of a regular triangulation (obtained from a finite difference grid, see figure 4.4) while the second type is an unstructured triangulation (see figure 4.5). The characteristics of the unstructured triangulations are given in table 4.2. The meshes RS2 and RS3 (respectively, meshes RU2 and RU3) have been obtained by uniform divisions of mesh RS1 (respectively, mesh RU1). For both geometries, the initialization is given by a uniform flow characterized by $\rho_0 = 1$, $u_0 = 1$, $v_0 = 0$ and $p_0 = \dfrac{1}{\gamma_e M^2}$ where M denotes the free-stream Mach number. For the first geometry, a slip condition is applied on the horizontal sides while an inflow (respectively, outflow) condition is applied on the left (respectively, right) vertical side. For the `NACA0012` airfoil, three unstructured triangular meshes have been used whose characteristics are given in table 4.3 (see figure 4.6 for a partial view of mesh N1). Mesh N2 and N3 have been obtained by uniform divisions of mesh N1.

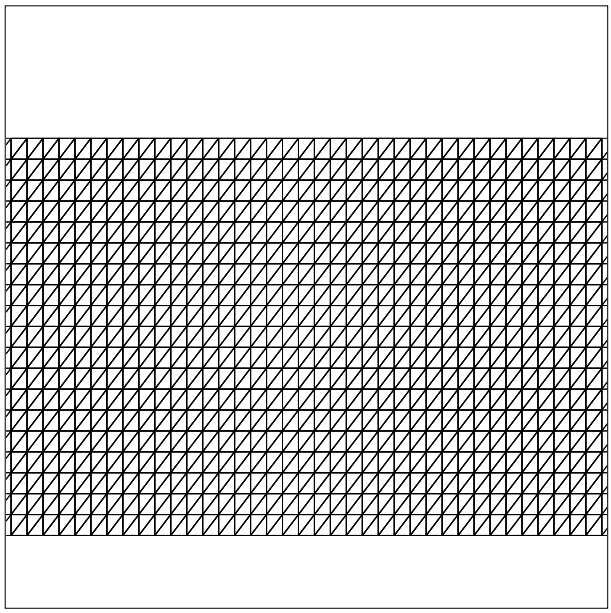


Figure 4.4: Structured triangular mesh of a rectangular domain

Table 4.1: Characteristics of the regular triangular meshes for the rectangular domain

Mesh	# Vertices	# Triangles	# Edges
RS1	4,000	7,562	11,561
RS2	16,000	31,442	47,441
RS3	64,000	126,962	190,961

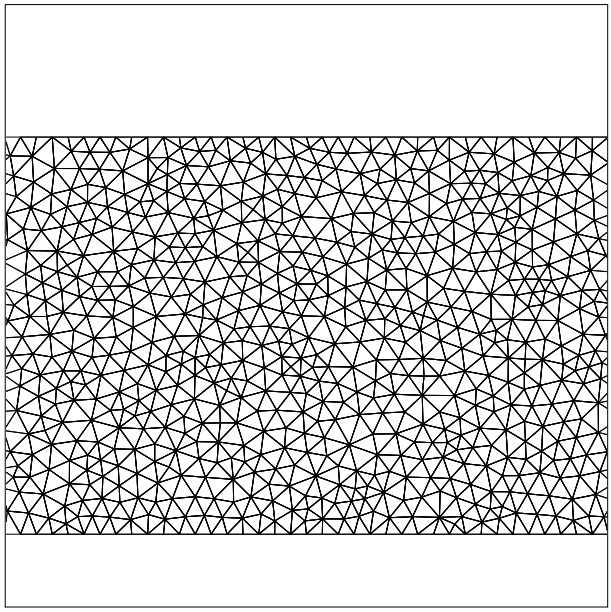


Figure 4.5: Unstructured triangular mesh of a rectangular domain

Table 4.2: Characteristics of the unstructured meshes for the rectangular domain

Mesh	# Vertices	# Triangles	# Edges
RU1	3,740	7,041	10,780
RU2	14,520	28,164	42,683
RU3	57,203	112,656	169,858

Table 4.3: Characteristics of the meshes for the `NACA0012` airfoil

Mesh	# Vertices	# Triangles	# Edges
N1	3,114	6,056	9,170
N2	12,284	24,224	36,508
N3	48,792	96,896	145,688

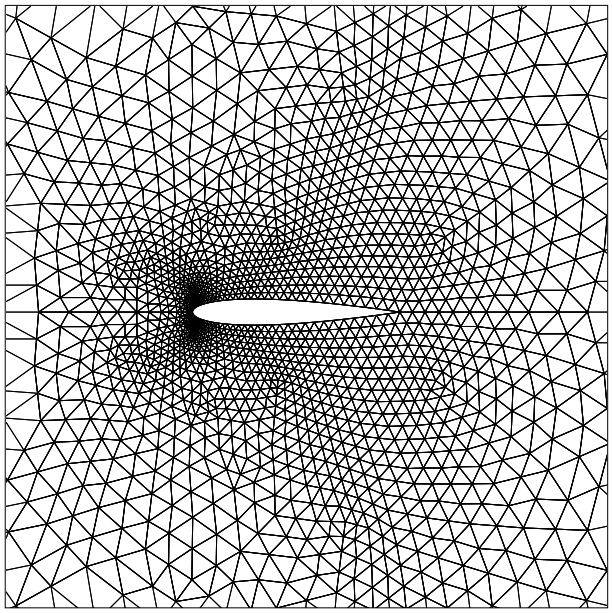


Figure 4.6: Unstructured triangular mesh around the NACA0012 airfoil

4.1.3.6.3 Flow inside a rectangular domain: regular triangulations. We consider the solution of the linear system resulting from the first time step for several flow conditions corresponding to values of the free-stream Mach number M ranging from 0.1 to 0.9 and using a two-subdomain decomposition of meshes RS1 to RS3 of table 4.1. The results are summarized on figure 4.7 where we have represented, for each mesh, the required number of Richardson iterations to reduce the initial normalized residual to the threshold $\varepsilon_i = 10^{-10}$. The linear threshold for the solution of the local systems has been set to $\varepsilon_l = 10^{-10}$. We observe that, as the mesh is refined from RS1 to RS3, the number of iterations increases slightly. This behavior is consistent with remark 2 of section 4.1.3. Moreover, the curves of figure 4.7 are in qualitative agreement with the theoretical behavior shown on figure 4.2. In the present case, the value $M = \sqrt{M_t^2 + M_n^2} = 0.6$ always yield the best convergence of the Schwarz algorithm. The analysis of section 4.1.3 as shown that a super-convergence behavior is obtained for a normal Mach number $M_n^\star = \dfrac{1}{3}$ with $M_t^\star = 0.$ where $M_n^\star$ and $M_t^\star$ respectively denote the tangential and normal Mach number at the interface which is supposed to be a straight line. Here, due to remark 4, the local tangential Mach Number is not equal to 0 in practice. Moreover, the local normal Mach number actually takes two values resulting from the fact that the interface Γ_{12} has a regular broken line shape since we make use of regular triangulations. For $M = 0.6$, these two values are respectively equal to 0.476 and 0.566 for all the meshes of table 4.1. We simply conclude this series of results by noting that, as the tangential Mach number increases from 0, figure 4.2 shows that the optimal value of the normal Mach number increases too resulting in a value of $M^\star = \sqrt{(M_t^\star)^2 + (M_n^\star)^2} > \dfrac{1}{3}$. Clearly, the optimal value of M obtained in practice is consistent with the results of section 4.1.3.

4.1.3.6.4 Flow inside a rectangular domain : unstructured triangulations. As in the previous series of numerical experiments, we consider the solution of the linear system resulting from the first time step for several flow conditions corresponding to values of the free-stream Mach number M ranging from 0.1 to 0.9 and using a two-subdomain decomposition of meshes RU1 to RU3 of table 4.2. The results are summarized on figure 4.8 where we show, for each mesh, the required number of Richardson iterations to reduce the initial normalized residual to the threshold $\varepsilon_i = 10^{-10}$. The linear threshold for the solution of the local systems has been set to $\varepsilon_l = 10^{-10}$. Clearly, this second series of experiments confirm the observations made when using the meshes

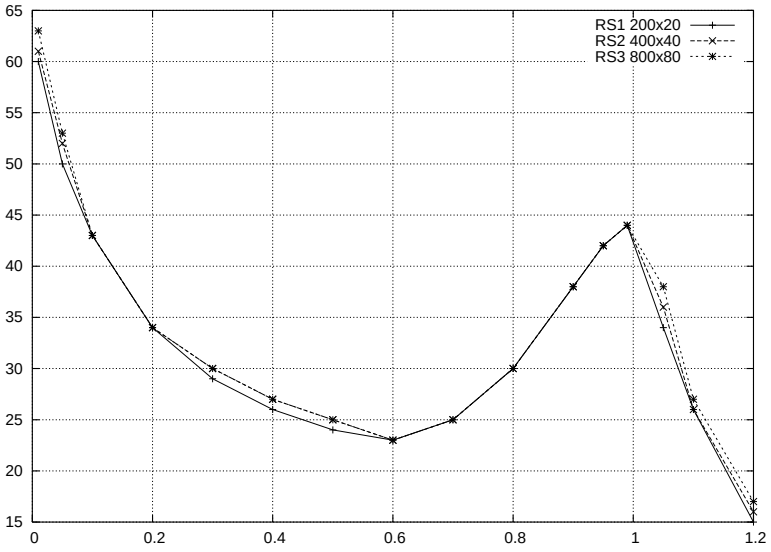


Figure 4.7: Convergence of the interface system
Flow inside a rectangular domain (regular triangulations) : 2 subdomain decomposition
X-axis : free-stream Mach number ($M = \sqrt{M_t^2 + M_n^2}$) - Y-axis : # Richardson iterations

based on regular triangulations. The value $M = \sqrt{M_t^2 + M_n^2} = 0.6$ is still the one that yield the best convergence of the Schwarz algorithm (in that case the normal Mach number values are in the range [0.240 , 0.600]).

4.1.3.6.5 Flow around a `NACA0012` airfoil. We conclude this section with results of numerical experiments involving the `NACA0012` geometry. Figure 4.9 summarizes the results obtained for several flow conditions corresponding to values of the free-stream Mach number ranging from 0.1 to 0.7 and using a two-subdomain decomposition of meshes N1 to N3 of table 4.3. As previously, the curves on this figure represent the required numbers of Richardson iterations to reduce the initial normalized residual to the threshold $\varepsilon_i = 10^{-10}$. The linear threshold for the solution of the local systems has been set to $\varepsilon_l = 10^{-10}$. This figure calls for several remarks:

- the consequences of using unstructured triangulations on the shape of the interface Γ are more remarkable here than for the flow calculations inside a rectangular domain since the mesh is refined in the vicinity of the airfoil. This time, the optimal number of Richardson iterations is essentially the same (that is, the best convergence of the Schwarz algorithm is observed) when $M \in [0.6\ ,\ 0.7]$ for each of the underlying meshes.
- For $M = 0.7$, the number of Richardson iterations is respectively equal to 63 and 89 for mesh N1 and N3 that is, this number increases by a factor 1.4 whereas the characteristic dimension of the triangulation has decreased by a factor 4. We can conclude that the additive Schwarz algorithm is slightly sensible to the refinement of the discretization as it was already the case with the previous test cases.
- The reader will note that we do not provide experimental values for flow calculations corresponding to a free-stream Mach number in the range [0.7 , 1.0]. As a matter of fact, for this range of values, the flow approaches the transonic regime and the non-linearity characterizing the resulting flow features tends to stiffen the implicit system. As a consequence, the Richardson method fails to converge. In that case, it is preferable to use a Krylov method such as GMRES for solving the interface system. This has been done but, since the other figures are given for the Richardson method, we decided to limit our presentation of results to the ones obtained with this method. Also, contrary to the previous test cases, even though the initial flow is uniform, the solution obtained at the end of the first time step is no more uniform due to the presence of the obstacle. In some sense, the constant coefficient theory developed in section 4.1.3 is

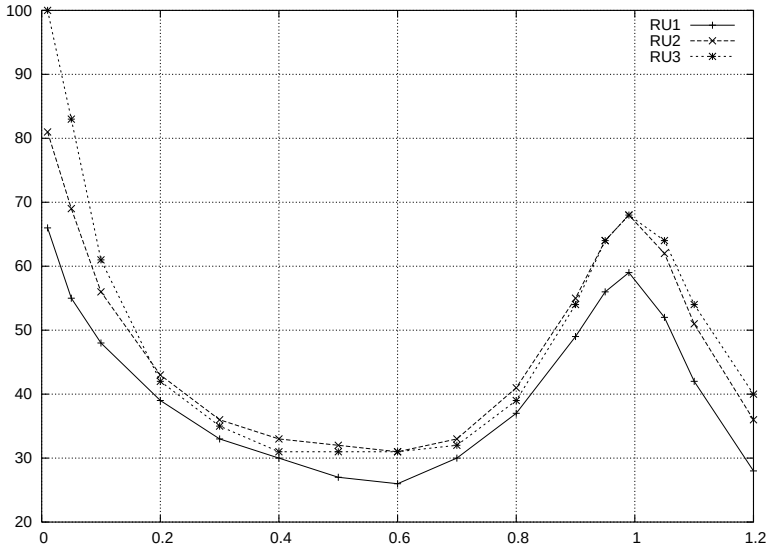


Figure 4.8: Convergence of the interface system
Flow inside a rectangular domain (unstructured triangulations) : 2 subdomain decomposition
X-axis : free-stream Mach number ($M = \sqrt{M_t^2 + M_n^2}$) - Y-axis : # Richardson iterations

questionable. For these reasons, the correlation between theoretical and experimental results is only partially demonstrated for the present test case.

4.1.4 Calculation of steady flows

In the first part of this section, we describe a particular implementation of the additive Schwarz algorithm (4.14) proposed in section 4.1.2 in the framework of the mixed element/volume formulation of section 1.2.1. The objective is to obtain a discrete variant of (4.14) for time advancing the solution of the Euler equations using the linearized implicit scheme described in section 1.4.1. In other words, we place ourselves in the context of section 4.1.2 since the additive Schwarz algorithm will be applied to the linearized Euler equations. Moreover, the discrete approach which is described here has two main distinctive features: on one hand, the resulting algorithm aims at solving an interface system acting on a reduced set of unknowns; on the other hand, the local systems that are obtained at each iteration of the algorithm are approximately solved using a multigrid method. In the second part of this section, we apply the resulting algorithm to the calculation of steady compressible flows modeled by the two-dimensional Euler equations.

4.1.4.1 Implementation of the additive Schwarz algorithm

4.1.4.1.1 Definition of the interface and formulation of the interface conditions. We recall that the parallelization strategy adopted in the original flow solver combines a partitioning of the domain and a message-passing programming model. In section 4.1.3.6.1 we motivated the use of a one-triangle wide overlapping region that is shared by neighboring subdomains. Then, it is easily seen that within this setting, the interface between two neighboring subdomains is a non-overlapping one from the viewpoint of the dual discretization of Ω in terms of control surfaces; if Ω_1 and Ω_2 are neighbors then:

$$\Gamma = \Omega_1 \cap \Omega_2 = \bigcup_{C_{1_k} \in \Omega_1, C_{2_k} \in \Omega_2} \partial C_{1_k} \cap \partial C_{2_k}$$

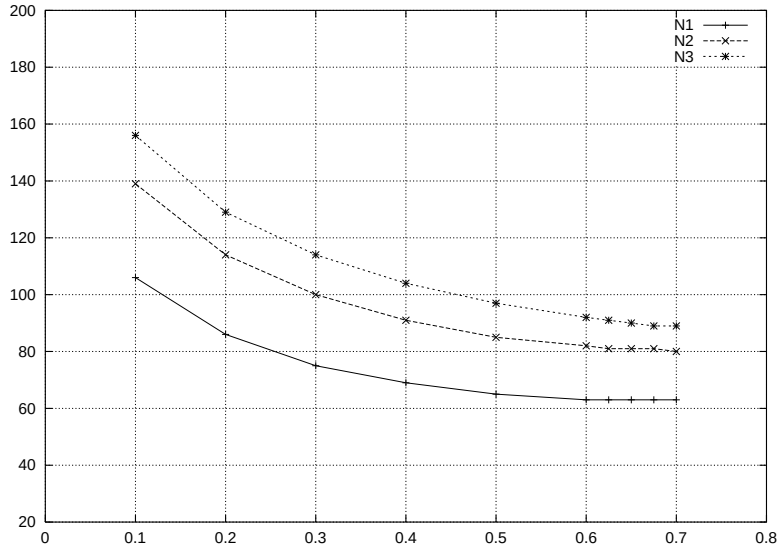


Figure 4.9: Convergence of the interface system
External flow around a NACA0012 airfoil : 2 subdomain decomposition
X-axis : free-stream Mach number ($M = M_t + M_n$) - Y-axis : # Richardson iterations

In order to be able to construct an interface system, we need to consider a preliminary step which consists in the introduction of a redundant variable at the interface between two control surfaces (see figure 4.10), following a strategy adopted by Clerc[32]. Our approach is however different from the one described in [32] for what concern the nature of this redundant variable since, as detailed below, the latter is here defined as the normal flux between two control surfaces belonging to different subdomains.

To simplify the presentation, let us consider the case of a decomposition of Ω into two subdomains. Let $[s_i, s_j]$ be and edge such that C_i (associated with s_i) and C_j (associated with s_j) belong to two neighboring subdomains. We note that in the context of the classical solution approach discussed in section 1.4.1, $[s_i, s_j]$ is an internal edge of the mesh. Thus, the corresponding elementary convective flux (1.12) is computed using the approximate Riemann solver of Roe[120]. Then, an additive Schwarz formulation is obtained by setting the following interface conditions that are taken into account in the integral formulation (1.10) locally in each subdomain:

$$\left\{ \begin{array}{rcl} A^{c,-}(\widetilde{W}_{ij}, \vec{\eta}_{ij}) W_i^{(k+1)} & = & A^{c,-}(\widetilde{W}_{ij}, \vec{\eta}_{ij}) W_j^{(k)} \\ A^{c,+}(\widetilde{W}_{ij}, \vec{\eta}_{ij}) W_j^{(k+1)} & = & A^{c,+}(\widetilde{W}_{ij}, \vec{\eta}_{ij}) W_i^{(k)} \end{array} \right. \tag{4.38}$$

where $\widetilde{W}_{ij}$ is given by eq. 1.16. Using the notation (1.15), we rewrite the interface conditions (4.38) as:

$$\mathcal{A}_{ij}^{-}(\widetilde{W}_{ij}) W_i^{(k+1)} = \mathcal{A}_{ij}^{-}(\widetilde{W}_{ij}) W_j^{(k)} \quad \text{and} \quad \mathcal{A}_{ij}^{+}(\widetilde{W}_{ij}) W_j^{(k+1)} = \mathcal{A}_{ij}^{+}(\widetilde{W}_{ij}) W_i^{(k)} \tag{4.39}$$

In the following, we show how the interface conditions (4.39) can be introduced in the linearization of the approximate Riemann solver of Roe (see eq. (1.86) and (1.87) of section 1.4.1) in order to obtain a new form of the global Jacobian matrix $P(W^n)$ (see eq. (1.84)) that will characterize the discrete variant of the additive Schwarz algorithm (4.14).

4.1.4.1.2 The discrete variant of the Schwarz algorithm. We introduce an auxiliary variable denoted by $W^\star$ (see figure 4.10) such that:

$$\left(\mathcal{A}_{ij}^{-}(\widetilde{W}_{ij}) W_j \right) |_{s_j} = \left(\mathcal{A}_{ij}^{-}(\widetilde{W}_{ij}) W^\star \right) |_{s_{ij}} \quad \text{and} \quad \left(\mathcal{A}_{ij}^{+}(\widetilde{W}_{ij}) W_i \right) |_{s_i} = \left(\mathcal{A}_{ij}^{+}(\widetilde{W}_{ij}) W^\star \right) |_{s_{ij}} \tag{4.40}$$

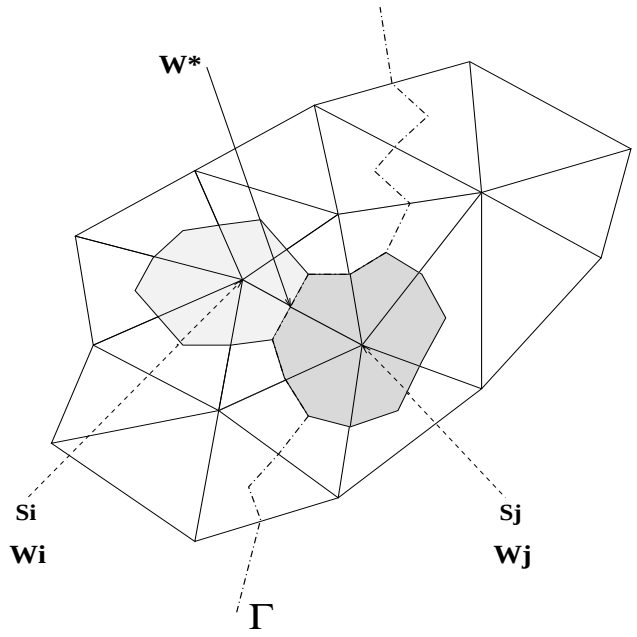


Figure 4.10: Definition of a redundant variable at the interface $\Gamma = \Omega_1 \cap \Omega_2$

with $s_{ij} = \dfrac{s_i + s_j}{2}$, and we define:

$$\begin{aligned} \Phi_{\Gamma,ij} = |\mathcal{A}_{ij}(\widetilde{W}_{ij})|W^\star &= \left(\mathcal{A}_{ij}^+(\widetilde{W}_{ij}) - \mathcal{A}_{ij}^-(\widetilde{W}_{ij})\right) W^\star \\ &= \mathcal{A}_{ij}^+(\widetilde{W}_{ij})W_i - \mathcal{A}_{ij}^-(\widetilde{W}_{ij})W_j \end{aligned} \tag{4.41}$$

In the present approach, $\Phi_{\Gamma,ij}$ is an interface unknown which is expressed as a normal flux associated to the edge $[s_i, s_j]$. Then, we can write:

$$\Phi_{\Gamma,ij} = \left(T(\widetilde{W}_{ij},\vec{\eta}_{ij})|\Lambda(\widetilde{W}_{ij},\vec{\eta}_{ij})|T^{-1}(\widetilde{W}_{ij},\vec{\eta}_{ij})\right) W^*$$

$$\Updownarrow$$

$$W^\star = \left(T(\widetilde{W}_{ij},\vec{\eta}_{ij})|\Lambda(\widetilde{W}_{ij},\vec{\eta}_{ij})|^{-1}T^{-1}(\widetilde{W}_{ij},\vec{\eta}_{ij})\right) \Phi_{\Gamma,ij}$$

The positive and negative parts of the interface flux $\Phi_{\Gamma,ij}$ are given by:

$$\begin{aligned} \Phi_{\Gamma,ij}^\pm = \mathcal{A}_{ij}^\pm(\widetilde{W}_{ij})W^* &= \left(T(\widetilde{W}_{ij},\vec{\eta}_{ij})\Lambda^\pm(\widetilde{W}_{ij},\vec{\eta}_{ij})T^{-1}(\widetilde{W}_{ij},\vec{\eta}_{ij})\right) W^* \\ &= \left(T(\widetilde{W}_{ij},\vec{\eta}_{ij})\Lambda^\pm(\widetilde{W}_{ij},\vec{\eta}_{ij})|\Lambda(\widetilde{W}_{ij},\vec{\eta}_{ij})|^{-1}T^{-1}(\widetilde{W}_{ij},\vec{\eta}_{ij})\right) \Phi_{\Gamma,ij} \end{aligned} \tag{4.42}$$

and can be written in condensed form as $\Phi_{\Gamma,ij}^\pm = P^\pm(\widetilde{W}_{ij})\Phi_{\Gamma,ij}$.

Moreover, the elementary linearized fluxes associated with the control surfaces C_i and C_j can be written as (see eq. (1.85) of section 1.4.1) :

$$\left\{\begin{array}{lcl} \Phi(W_i^n, W_j^n, W_i^{n+1}, W_j^{n+1}, \vec{\eta}_{ij}) & = & \Phi(W_i^n, W_j^n, \vec{\eta}_{ij}) \\ & + & \left(A^c(W_i^n, \eta_{ij}) - \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n)\right)\delta W_i^{n+1} + \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n)\delta W_j^{n+1} \\ \Phi(W_j^n, W_i^n, W_j^{n+1}, W_i^{n+1}, \vec{\eta}_{ji}) & = & -\Phi(W_i^n, W_j^n, \vec{\eta}_{ij}) \\ & - & \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n)\delta W_j^{n+1} - \left(A^c(W_i^n, \eta_{ij}) - \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n)\right)\delta W_i^{n+1} \end{array}\right. \tag{4.43}$$

By making the following approximation at the interface:

$$A^c(W_i^n, \eta_{ij}) - \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n) \cong \mathcal{A}_{ij}^+(\widetilde{W}_{ij}^n)$$

we can rewrite the right-hand side terms of (4.43) as:

$$\left\{\begin{array}{lcl} \mathsf{RHS}_{ij} & = & \Phi(W_i^n, W_j^n, \vec{\eta}_{ij}) + \left(A^c(W_i^n, \eta_{ij}) - \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n)\right)\delta W_i^{n+1} + \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n)\delta W^\star \\ & = & \Phi(W_i^n, W_j^n, \vec{\eta}_{ij}) + \left(A^c(W_i^n, \eta_{ij}) - \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n)\right)\delta W_i^{n+1} + P^-(\widetilde{W}_{ij})\delta\Phi_{\Gamma,ij} \\ \mathsf{RHS}_{ji} & = & -\Phi(W_i^n, W_j^n, \vec{\eta}_{ij}) - \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n)\delta W_j^{n+1} - \mathcal{A}_{ij}^+(\widetilde{W}_{ij}^n)\delta W^\star \\ & = & -\Phi(W_i^n, W_j^n, \vec{\eta}_{ij}) - \mathcal{A}_{ij}^-(\widetilde{W}_{ij}^n)\delta W_j^{n+1} - P^+(\widetilde{W}_{ij})\delta\Phi_{\Gamma,ij} \end{array}\right. \tag{4.44}$$

Putting together eq. (4.43) for purely internal edges and eq. (4.44) for interface edges we can form a global linear system that characterizes the discrete variant of the additive Schwarz algorithm (4.14):

$$\left(\begin{array}{ccc} \mathcal{M}_1 & 0 & \mathcal{M}_{1\Gamma} \\ 0 & \mathcal{M}_2 & \mathcal{M}_{2\Gamma} \\ \mathcal{F}_{1\Gamma} & \mathcal{F}_{2\Gamma} & \mathsf{Id} \end{array}\right)\left(\begin{array}{c} \delta W_1^{n+1} \\ \delta W_2^{n+1} \\ \delta\Phi_\Gamma \end{array}\right) = \left(\begin{array}{c} b_1 \\ b_2 \\ 0 \end{array}\right) \tag{4.45}$$

where we have made a distinction between the vectors of purely interior unknowns associated to subdomains Ω_1 and Ω_2 (that is δW_1^{n+1} and δW_2^{n+1}) and the vector of interface unknowns represented by $\delta\Phi_\Gamma$. Moreover, $\mathcal{M}_1$ (respectively $\mathcal{M}_2$) is the matrix that couples the unknowns associated with vertices internal to Ω_1 (respectively Ω_2) whereas $\mathcal{F}_{1\Gamma}$, $\mathcal{F}_{2\Gamma}$, $\mathcal{M}_{1\Gamma}$ and $\mathcal{M}_{2\Gamma}$ are coupling matrices between internal and interface unknowns. These matrix operators are detailed in [39]. At this point, the internal unknowns can be eliminated in favor of the interface ones by applying a sub-structuring technique to system (4.45) yielding the following interface system:

$$\begin{array}{rcl} S\delta\Phi_\Gamma & = & [\mathsf{Id} - (\mathcal{F}_{1\Gamma}\mathcal{M}_1^{-1}\mathcal{M}_{1\Gamma} + \mathcal{F}_{2\Gamma}\mathcal{M}_2^{-1}\mathcal{M}_{2\Gamma})]\delta\Phi_\Gamma \quad = \quad g \\ & & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad = \quad -[\mathcal{F}_{1\Gamma}\mathcal{M}_1^{-1}b_1 + \mathcal{F}_{2\Gamma}\mathcal{M}_2^{-1}b_2] \end{array} \tag{4.46}$$

As usual in this context, once system (4.46) has been solved for $\delta\Phi_\Gamma$, one obtain the values of the purely internal unknowns by performing independent local solves :

$$\left\{\begin{array}{lcl} \delta W_1^{n+1} & = & \mathcal{M}_1^{-1}(b_1 - \mathcal{M}_{1\Gamma}\delta\Phi_\Gamma) \\ \delta W_2^{n+1} & = & \mathcal{M}_2^{-1}(b_2 - \mathcal{M}_{2\Gamma}\delta\Phi_\Gamma) \end{array}\right. \tag{4.47}$$

Note that, in practice, the interface operator S in never formed. Instead, since a Krylov type method such as GMRES[124] is used to solve system (4.46), only the matrix/vector product involving S needs to be computed. Again, this requires solving local systems with the matrix operators $\mathcal{M}_1$ and $\mathcal{M}_2$.

Remark 5 *It is worthwhile to note that the vector of interface unknowns $\delta\Phi_\Gamma$ (or $\delta W^\star$) is only introduced for the formation of the global linear system (4.45) that is, it does not define an additional set of unknowns for the overall problem. This justifies the notation $\delta\Phi_\Gamma$ instead of $\delta\Phi_\Gamma^{n+1}$.*

4.1.4.2 Solution strategy for the local problems

As mentioned above, eq. (4.47) defines independent linear system solves in each subdomain. In the domain decomposition framework and especially for CSM (Computational Structural Mechanics) applications, these local problems are generally solved exactly using a factorization method which is also the reference solver when a global approach is adopted. The situation is somewhat different for most CFD applications since, in practice, the global linear system associated to the Jacobian matrix $P(W^n)$ (see eq. (1.84)) is solved iteratively and with a low accuracy. In the present study, we have decided to adopt the same approach for local linear system solves of eq. (4.47). In particular, we are interested in studying (at least experimentally) the influence of approximate local solutions on the convergence of the interface system solver (that is GMRES) as well as on the convergence of the overall domain decomposed flow solver. We note that the robustness of Krylov methods such as GMRES, with respect to inexact matrix-vector products has been the subject of several investigations; see for example Bouras and Fraysse[12] and also Bouras *et al.*[13] for a discussion of that point in the context of Schur complement type domain decomposition methods. A rigorous theory of inexact Krylov subspace methods has been proposed in a recent paper from Simoncini and Szyld[129]. We note that, in the present study, we do not make use of a specific stopping criteria for monitoring the accuracy of the computed matrix-vector products such as the one proposed in [129] since, as will be seen in the results section, the underlying linear systems are solved with a low accuracy. In this study, a linear multigrid strategy applied at the subdomain level has been adopted for the local solutions. The smoother is a point-wise Gauss-Seidel method. The method is described in more details in section 3.1 of chapter 3. Its main features are recalled below.

- **Grid coarsening by agglomeration**. The coarsening strategy is based on the use of macro elements (macro control surfaces) which form the coarse discretizations of the computational domain. Starting from a fine unstructured triangulation, one wants to generate a hierarchy of coarse levels; this can be achieved using a "greedy" type coarsening algorithm that assembles neighboring control volumes of the finest grid to build the macro elements of the coarser level.
- **Coarse grid approximation for convective terms**. The convective fluxes are integrated between two control volumes of the finest mesh; they are computed in the same way on a coarse level, between two macro elements. However, on the coarse grids, this computation is limited to first order accuracy because nodal gradients cannot be evaluated as they are on a fine mesh; this is really not a problem here as the multigrid method is used to accelerate the solution of a linear system whose Jacobian matrix is based on the linearization of a first order convective flux.
- **Coarse grid approximation for diffusive terms**. To evaluate the diffusive terms on a coarse level, related basis functions are needed. Indeed, in the finite element formulation on the fine grid, the equations are integrated and assembled by edges (convective terms) and triangles (diffusive terms). As triangles do not exist on the coarser grids, it is necessary to define a new formulation for the calculation of diffusive terms.
- **Inter-grid transfer operators**. The solution restriction operator is constructed as a weighted approximation of fine grid components while the right-hand side restriction operator is obtained by a summation of fine grid components. Finally, the prolongation operator is a trivial injection of coarse grid components.

4.1.4.3 Numerical results

In this section we apply the domain decomposition algorithm proposed in section 4.1.4.1 to the calculation of steady inviscid flows and make some comparisons with the global approach described in section 1.4.1 of chapter 1. The test cases under consideration correspond to the calculation of external flows around a `NACA0012` airfoil (see figure 4.6). In addition to the unstructured meshes given in table 4.3, a finer discretization has been

considered: mesh N4 contains 194,480 vertices, 387,584 elements and 582,064 edges. The following situations have been considered :

S1 : the subsonic flow at a free stream Mach number M_∞ equal to 0.3 and an angle of attack of 0°. In that case, the extension to second order accuracy in space does not make use of a limiter. The time step is obtained using constant values of the CFL number. The value CFL=1000 has been used for the steady flow computation however, as will be seen in the sequel, we have also investigated the influence of the value of the CFL number on the convergence of the domain decomposition solver.

S2 : the transonic flow at a free stream Mach M_∞ number equal to 0.85 and an angle of attack of 0°. In that case, the time step is obtained using the law CFL=$5 \times k_t$ where k_t denotes the time iteration. The Van Albada limiter is used in the MUSCL technique (see Fezoui and Dervieux[52] for more details).

In each case, the calculation starts from a uniform flow characterized by $\rho_0 = 1$, $u_0 = 1$, $v_0 = 0$ and $p_0 = \dfrac{1}{\gamma_e M_\infty^2}$. The results presented below are all characterized by the fact that the local systems induced by the domain decomposition solver are solved approximately. The steady solutions corresponding to these two test cases are shown on figure 4.11 in the form of steady contour lines of the Mach number for calculations based on mesh N3.

4.1.4.3.1 Computing platforms and conventions. Numerical experiments have been performed on a cluster of 216 HP e-vectra nodes (Pentium III/733 Mhz with 256 Mb of SDRAM 133 Mhz each) interconnected by several Ethernet 100 Mbit/s switches. The MPI implementation is MPICH. The code is written in Fortran 77 and the GNU G77 compiler has been used with maximal optimization options. Performance results are given for 64 bit arithmetic computations. In the following tables, N_p is the number of processes for the parallel execution (N_p also represents the number of subdomains); N_g is the total the number of grid levels (including the finest grid level) used in the multigrid solver for local systems; "$\#\ i_t$" is the required number of pseudo-time steps to reach the steady state (convergence to the steady state is monitored using the normalized energy residual); "Total time" denotes the total (elapsed) execution time and "CPU time" denotes the total CPU time (taken as the maximum value of the local per process measures); "% CPU" denotes the ratio of "CPU time" to "Total time". This ratio is our principal measure of parallel efficiency. The difference between "Total time" and "CPU time" basically yields the sum of the communication and idle times, the latter being related to computational load unbalance. The parallel speedup $S(N_p)$ is always calculated using the elapsed execution times. Finally, the term "linear threshold" is used to characterize the accuracy of the linear system solves (i.e. the level of reduction of the initial linear residual).

In the following, we consider and compare the performances of two parallel solution strategies:

- the global solution strategy (see section 1.4.1 of chapter 1) where the implicit system of eq. (1.84) is solved using Jacobi relaxations (the corresponding linear threshold is denoted by ε_g);
- the hybrid domain decomposition/multigrid method introduced in section 4.1.4.1 where the GMRES method is used to solve the interface system (4.46) (the corresponding linear threshold is denoted by ε_{ddm}). In this approach, the multigrid by agglomeration domain is used to solve the linear system obtained at each iteration of the GMRES method (the corresponding linear threshold is denoted by ε_{mg}). A point-wise Gauss-Seidel relaxation method is playing the role of the smoother in a multigrid V-cycle.

4.1.4.3.2 Performance results for the S1 test case. The solution methods differ through the strategy used for solving the linear systems obtained at each time step:

- **GLOB.a** : the global solution strategy with $\varepsilon_g = 10^{-1}$;
- **DDM/MG.a** : the hybrid domain decomposition/multigrid with $\varepsilon_{\text{ddm}} = 10^{-1}$. The local linear systems obtained at each GMRES iteration are (approximately) solved using several V(2,2,2) multigrid cycles with $\varepsilon_{\text{mg}} = 10^{-1}$.

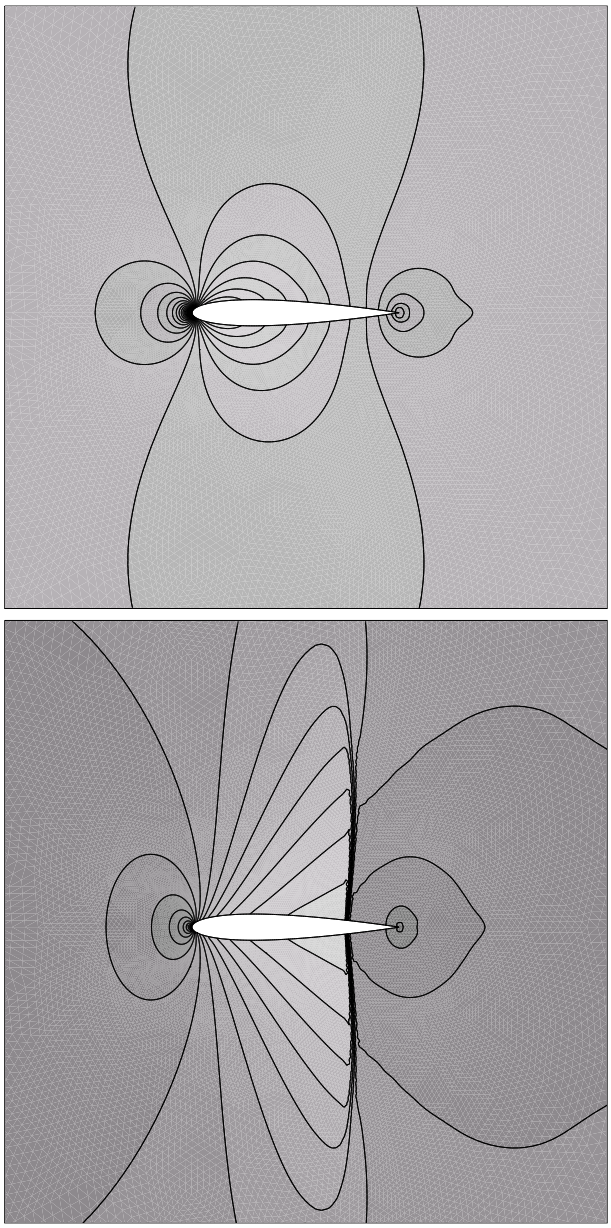


Figure 4.11: Inviscid flow around a `NACA0012` airfoil
Steady contour lines of the Mach number: S1 (top) and S2 (bottom) test cases

- **DDM/MG.b** : the **DDM/MG.a** strategy where the local systems are solved using only 2 V(2,2,2) multigrid cycle, all other components and parameters being unchanged.

The effective numbers of time steps to reach the steady state (initial normalized energy residual reduced by a factor 10^6) and corresponding execution times are given in table 4.4 for calculations based on the finest mesh (i.e. mesh N4). The reference result is given by the convergence to steady state of the global solution strategy based on the Jacobi relaxation method **GLOB.a** (141 time steps independently of the number of subdomains). We note that reducing the linear threshold from $\varepsilon_g = 10^{-1}$ to $\varepsilon_g = 10^{-2}$ did not result in a noticeable reduction of the number of time steps to reach the steady state but required larger execution times. The hybrid domain decomposition/multigrid strategy **DDM/MG.a** yields essentially the same number of time steps with a slight increase for $N_p = 64$. Obtaining exactly the same number of time steps as the one resulting from the application of the **GLOB.a** strategy would require reducing by one order of magnitude the values of the linear thresholds ε_{ddm} and ε_{mg} at the expense of a large increase of the execution times which is clearly not a viable option. Nevertheless, the solution strategy **DDM/MG.a** is the one demonstrating the best parallel performances which is of course related to the fact that this method has the most favorable computation to communication ratio of the three methods tested. We note that the hybrid domain decomposition/multigrid strategy **DDM/MG.b** based on a fixed (and minimal) number of V(2,2,2) cycles for the solution of the local linear systems (i.e. strategy **DDM/MG.b**) is characterized by a number of time steps which is more sensitive to the number of subdomains and always higher than the one characterizing the reference solution (from 147 to 155 depending on the value of N_p instead of 141). Despite this fact, this solution strategy yields a reduction by a factor 2.0 of the total simulation time as compared to the reference strategy **GLOB.a**.

We conclude this series of results with a scalability analysis of the solution strategy **DDM/MG.a**. This kind of performance assessment is particularly important for domain decomposition methods whose convergence rate generally depends on two parameters: the characteristic size h of the computational mesh (which is directly related to the global number of degrees of freedom) and the characteristic size H of the macro-mesh associated to the partitioning of the computational domain (which is related to the number of subdomains N_p). Table 4.5 summarizes the results of the application of the solution strategy **DDM/MG.a** to meshes N2 to N4 and for a number of subdomains increasing from 8 by the same factor as the refinement ratio between two consecutive meshes (4 in the present case). The increase of the number of pseudo-time steps required to obtain the steady state solution (non-linear convergence) is a natural consequence of the refinement of the mesh. Therefore, we have reported in this table, the average CPU time and average total time for one pseudo-time step. In addition, figure 4.12 shows the evolution of the effective number of GMRES iterations for solving the interface system (4.46) versus the pseudo-time step number. When switching from $N_p = 8$ (respectively, mesh N2) to $N_p = 128$ (respectively, mesh N4), the average CPU time per pseudo-time step increases by 25.5% while the corresponding figure for the total time is 91%. Figure 4.12 shows that the conditioning of the interface system is weakly dependent on both h and H. However, this remark has to be moderated by the fact that, in the present case, the interface systems are solved with a low accuracy since the corresponding linear threshold has been fixed $\varepsilon_{\text{ddm}} = 10^{-1}$. Nevertheless, we can state that the increase in CPU time when switching from $N_p = 8$ to $N_p = 128$ subdomains is essentially due to the increase in the number of GMRES iterations. For what concern the total CPU time, the observed degradation can be related to two factors that classically impact parallel performances of numerical algorithms:

- communication times: the interconnection network of the underlying computing platform (i.e. Ethernet 100 Mbit/s) is rather standard and, at the time of this study, probably the less efficient one among other possibilities. The solution strategy considered here is characterized by a reduced communication load as compared, for instance, to the global solution strategy **GLOB.a**. As already noticed, this favorable behavior is well illustrated by the results reported in table 4.4 for computations based on mesh N4. However, for large values of N_p (i.e. $N_p > 64$) the parallel efficiency of the domain decomposition solvers **DDM/MG.a** and **DDM/MG.b** remains rather low. It is clear that this rapid degradation of the parallel efficiency (especially for $N_p = 128$) is for a great part due to the poor communication performances of the interconnection network.

- idle times: in the present case, the main source of idle times is found in the local system solution phase. Indeed, since in the solution strategy **DDM/MG.a** the resolution of the local systems is characterized

by a linear threshold $\varepsilon_{\text{mg}} = 10^{-1}$, we can reasonably expect that the effective number of multigrid cycles demonstrates slight variations from one subdomain to another, depending on the conditioning of the local problems. This necessarily incurs synchronization points at each GMRES iteration when computing the matrix/vector product $S\delta\Phi_\Gamma$ of eq. 4.46.

Table 4.4: S1 test case - timings for the steady state calculation (mesh N4)
Global solution strategy (parallel Jacobi linear solver) versus DDM/MG strategy (full GMRES)

Method	N_p	N_g	# i_t	CPU time	Total time	% CPU	$S(N_p)$
GLOB.a	32	1	141	661 sec	884 sec	75.0	1.00
-	64	1	141	351 sec	614 sec	57.0	1.45
-	128	1	141	231 sec	551 sec	42.0	1.60
DDM/MG.a	32	5	142	715 sec	872 sec	82.0	1.00
-	64	4	145	368 sec	502 sec	73.5	1.75
-	128	3	142	213 sec	338 sec	63.0	2.60
DDM/MG.b	32	5	147	484 sec	581 sec	83.5	1.00
-	64	4	155	291 sec	406 sec	72.0	1.45
-	128	3	152	147 sec	275 sec	53.5	2.15

Table 4.5: S1 test case - scalability analysis using method **DDM/MG.a**

Mesh	N_p	N_g	# i_t	CPU time per time step	Total time per time step	% CPU
N2	8	3	98	1.12 sec	1.25 sec	89.5
N3	32	3	102	1.27 sec	1.65 sec	77.0
N4	128	3	142	1.50 sec	2.40 sec	63.0

4.1.4.3.3 Performance results for the S2 test case. As with the S1 test case, the solution methods differ through the strategy used for solving the linear systems obtained at each time step:

- **GLOB.a** : the global solution strategy with $\varepsilon_g = 10^{-2}$;

- **DDM/MG.a** : the hybrid domain decomposition/multigrid with $\varepsilon_{\text{ddm}} = 10^{-1}$. The local linear systems obtained at each GMRES iteration are (approximately) solved using several V(4,4,4) multigrid cycles with $\varepsilon_{\text{mg}} = 10^{-1}$.

- **DDM/MG.b** : the **DDM/MG.a** strategy where the local systems are solved using only 1 V(4,4,4) multigrid cycle, all other components and parameters being unchanged.

- **DDM/MG.c** : the **DDM/MG.a** strategy where the local systems are solved using only 2 V(4,4,4) multigrid cycle, all other components and parameters being unchanged.

The effective numbers of time steps to reach the steady state (initial normalized energy residual reduced by a factor 10^6) and corresponding execution times are given in table 4.6 for calculations based on the finest mesh (i.e. mesh N4). Figure 4.13 shows the convergence profiles to steady-state for all the methods considered here using mesh N4 and for $N_p = 128$. The reference result is given by the convergence to steady state of the global solution strategy based on the Jacobi relaxation method **GLOB.a** (275 time steps independently of the

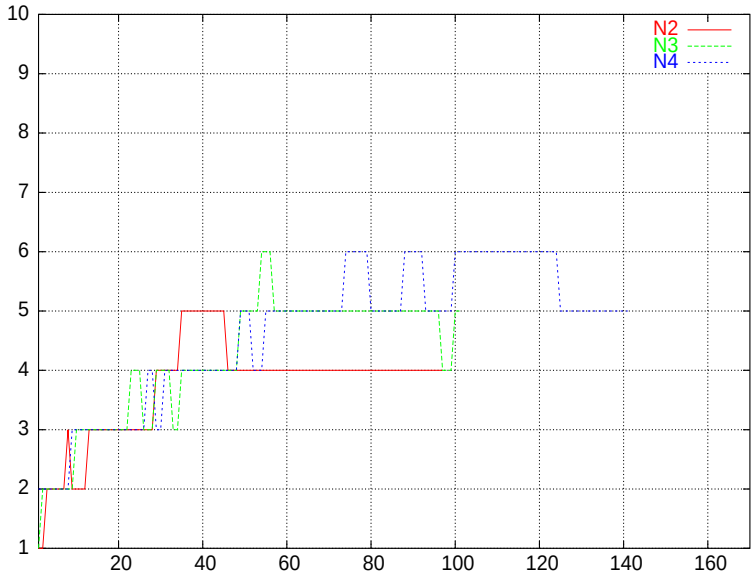


Figure 4.12: S1 test case - scalability analysis of solution strategy **DDM/MG.a**
Effective number of GMRES iterations for solving the interface system

number of subdomains). We note that, contrary to the S1 test case, a value $\varepsilon_g = 10^{-1}$ of the linear threshold resulted in number of pseudo-time steps and resulting total simulation times notably larger than those obtained for $\varepsilon_g = 10^{-2}$. For the domain decomposition solvers, a value $\varepsilon_{\text{mg}} = 10^{-1}$ has been found to be sufficient for obtaining comparable and, in some cases, faster convergence to the steady state. Similarly to the S1 test case, the domain decomposition solver based on a minimal, constant complexity strategy for the solution of local systems **DDM/MG.b** consistently outperforms the global solution strategy **GLOB.a**. For $N_p = 128$, this method yields a reduction of the total simulation time by a factor 2.1 as compared to the reference strategy **GLOB.a**. Results of a scalability analysis of the solution strategy **DDM/MG.a** are reported in table 4.7 and figure 4.14 leading to conclusions identical to those drawn for the S1 test case.

Table 4.6: S2 test case - timings for the steady state calculation (mesh N4)
Global solution strategy (parallel Jacobi linear solver) versus DDM/MG strategy (full GMRES)

Method	N_p	N_g	# i_t	CPU time	Total time	% CPU	$S(N_p)$
GLOB.a	32	1	275	1090 sec	1479 sec	74.0	1.00
-	64	1	275	588 sec	1001 sec	59.0	1.45
-	128	1	275	385 sec	910 sec	42.5	1.65
DDM/MG.a	32	5	258	1560 sec	1907 sec	82.0	1.00
-	64	4	268	870 sec	1060 sec	82.0	1.80
-	128	3	280	444 sec	676 sec	66.0	2.85
DDM/MG.b	32	5	291	867 sec	1047 sec	83.0	1.00
-	64	4	298	463 sec	660 sec	70.0	1.60
-	128	3	304	246 sec	431 sec	57.0	2.45
DDM/MG.c	32	5	262	1399 sec	1600 sec	87.5	1.00
-	64	4	270	736 sec	957 sec	77.0	1.70
-	128	3	281	356 sec	574 sec	62.0	2.80

Table 4.7: S2 test case - scalability analysis using method **DDM/MG.a**

Mesh	N_p	N_g	# i_t	CPU time per time step	Total time per time step	% CPU
N2	8	3	138	1.13 sec	1.26 sec	89.0
N3	32	3	187	1.33 sec	1.70 sec	78.5
N4	128	3	280	1.58 sec	2.41 sec	66.0

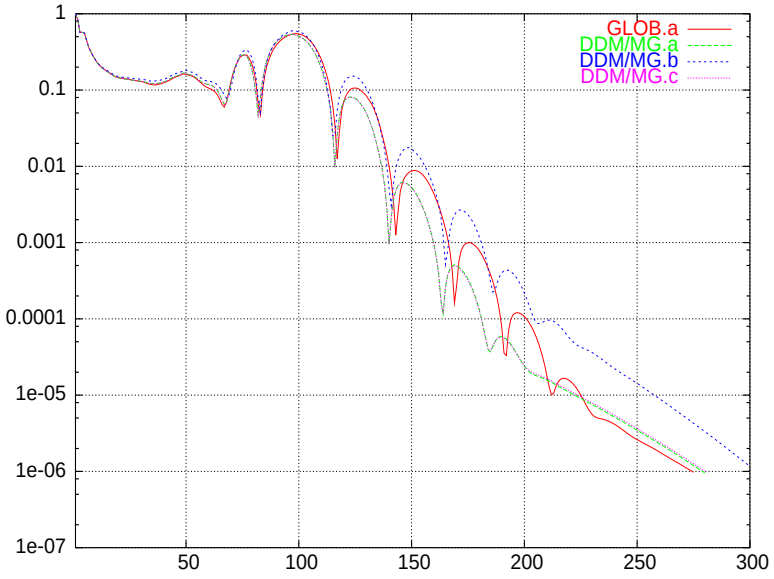


Figure 4.13: S2 test case (mesh N4) - non-linear convergence to the steady state
Normalized energy residual (log scale) versus pseudo-time step number

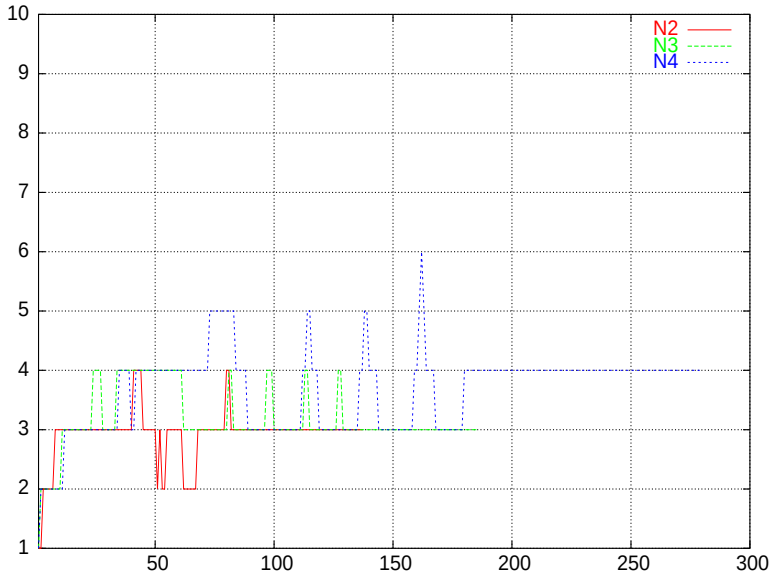


Figure 4.14: S2 test case - scalability analysis of solution strategy **DDM/MG.a**
Effective number of GMRES iterations for solving the interface system

4.2 Domain decomposition for laminar viscous flows

The objective of the present section is to solve the Navier-Stokes equations modeling compressible viscous flows by a non-overlapping domain decomposition method. The method presented here extends the one proposed in section 4.1 for the solution of the Euler equations. It relies on the formulation of an additive Schwarz type algorithm where the interface conditions express the continuity of normal flux components. As in section 4.1.4, the Navier-Stokes equations are discretized in space using the mixed element/volume formulation on unstructured triangular meshes described in section 1.2.1. Time integration of the semi-discrete equations relies on a linearized implicit scheme. However, in this section we will consider both steady and unsteady laminar viscous flows. For that purpose, we will used the time integration schemes described in sections 1.4.1 and 1.4.2. In each case, the non-overlapping domain decomposition algorithm is used for advancing the solution at each implicit time step or sub-time step. Algebraically speaking, the additive Schwarz algorithm is equivalent to a Jacobi iteration on a linear system whose matrix has a block structure. A sub-structuring technique can be applied to this matrix in order to obtain a fully implicit scheme in terms of interface unknowns. In the approach proposed here, the interface unknowns are numerical fluxes; more precisely, the vector of interface unknowns is composed, on one hand, of discrete convective fluxes computed on interface edges using the approximate Riemann solver of Roe[120] and, on the other hand, of discrete diffusive fluxes computed on interface triangles.

4.2.1 Additive Schwarz algorithm for the Navier-Stokes equations

The formulation of a domain decomposition algorithm in the continuous case is inspired from Quarteroni and Stolcis[116]. Assume that the computational domain $\Omega \in \mathbb{R}^2$ is decomposed into a set of non-overlapping subdomains Ω_i with $\Omega_i \cap \Omega_j = \Gamma_{ij}$ if Ω_j is a neighboring subdomain of Ω_i. Let W_i be the restriction of the solution of the Navier-Stokes equations (1.3) (see section 1.1 of chapter 1) to subdomain Ω_i. Then, W_i is the solution of the following BVP problem:

$$\left\{ \begin{array}{ll} \dfrac{\partial W_i}{\partial t} + \vec{\nabla}.\vec{\mathbb{F}}^c(W_i) = \dfrac{1}{\mathsf{Re}}\vec{\nabla}.\vec{\mathbb{F}}^v(W_i) & \text{for} \quad \vec{x} \in \Omega_i \\ \left[\left(\vec{\mathbb{F}}^c(W) - \dfrac{1}{\mathsf{Re}}\vec{\mathbb{F}}^v(W) \right).\vec{n}_{ij} \right]_{\Gamma_{ij}} = 0 & \text{for} \quad \vec{x} \in \Gamma_{ij} \\ \text{Boundary conditions on} \quad \partial\Omega \cap \partial\Omega_i & \end{array} \right. \tag{4.48}$$

where $\vec{\mathbb{F}}^c(W) = \left(F_x^c(W)\ ,\ F_y^c(W)\right)^T$ and $\vec{\mathbb{F}}^v(W) = \left(F_x^v(W)\ ,\ F_y^v(W)\right)^T$; $\vec{n}_{ij}$ denotes the normal vector at every point of Γ_{ij} directed from Ω_i to Ω_j. Moreover, $[a]_{\Gamma_{ij}}$ stands for the jump of the quantity a at the interface Γ_{ij} that is $[a]_{\Gamma_{ij}} = a\mid_{\Gamma_{ij}^+} - a\mid_{\Gamma_{ij}^-}$ where Γ_{ij}^+ and Γ_{ij}^- respectively denote the right and left sides of the interface Γ_{ij}. The interface condition is valid when Ω_j is a neighboring subdomain of Ω_i; it expresses the continuity of normal fluxes. This flux matching property is a natural consequence of the fact that the variable W is the weak solution of eq. (1.3) and so are the local solutions W_i.

As in section 4.1.4, we formulate an additive Schwarz algorithm which is applied to the linearized Navier-Stokes equations. Then, the goal is to devise a strategy for implementing the interface condition of eq. (4.48) in the context of the mixed element/volume formulation on unstructured triangular meshes of section 1.2.1 and, to deduce a new form of the global Jacobian matrix $P(W^n)$ (see eq. (1.84)) that will characterize the discrete variant of the additive Schwarz algorithm. This is the subject of the next section.

4.2.2 Implementation of the domain decomposition algorithm

In the discrete case, the definition of the interface is unchanged from section 4.1.4.1.1. We recall that the parallelization strategy adopted in the original flow solver combines a partitioning of the domain and a message-passing programming model. The partitioning of the domain is characterized by a one-triangle wide overlapping region that is shared by neighboring subdomains. For the implementation of the additive Schwarz algorithm, the main difficulty that we have to deal with is related to the fact that, in the mixed element/volume formulation,

convective fluxes are computed edgewise (see eq. (1.12)) while diffusive fluxes are evaluated element-wise (see eq. (1.22)). Indeed, a discretization method based on a finite volume formulation for both the convective and the diffusive fluxes, such as the one proposed by Rostand and Stoufflet[123], would have been more appropriate for the realization of this task. Here, in order to overcome this difficulty, we have adopted a strategy which consists in treating separately the convective and diffusive parts of the interface condition of eq. (4.48). To simplify the presentation, let us consider the case of a decomposition of Ω into two subdomains Ω_1 and Ω_2.

4.2.2.1 Continuity of the convective fluxes

Let $[s_i, s_j]$ be and edge such that C_i (associated with s_i) and C_j (associated with s_j) belong to two neighboring subdomains. We recall that in the mixed element/volume formulation, the corresponding elementary convective flux (1.12) is computed using the approximate Riemann solver of Roe[120]. Then, the convective part of the interface condition of eq. (4.48) is equivalent to setting:

$$\left\{ \begin{array}{rcl} A^{c,-}(\widetilde{W}_{ij}, \vec{\eta}_{ij}) W_i^{(k+1)} & = & A^{c,-}(\widetilde{W}_{ij}, \vec{\eta}_{ij}) W_j^{(k)} \\ A^{c,+}(\widetilde{W}_{ij}, \vec{\eta}_{ij}) W_j^{(k+1)} & = & A^{c,+}(\widetilde{W}_{ij}, \vec{\eta}_{ij}) W_i^{(k)} \end{array} \right. \tag{4.49}$$

at the interface Γ_{ij} ($\widetilde{W}_{ij}$ is given by eq. 1.16). The strategy used to exploit the conditions (4.49) in the formulation of the discrete Schwarz algorithm is exactly the same than the one used for the Euler equations (see section 4.1.4.1.2) thus yielding the definition of the vector of interface unknowns $\delta\Phi_\Gamma$.

4.2.2.2 Continuity of the diffusive fluxes

The goal of this section if to construct the interface unknowns associated with the diffusive or viscous fluxes. Since an elementary diffusive flux is computed on a triangle (see eq. (1.22)) and also to facilitate the implementation, we have chosen to impose the continuity of the whole diffusive flux components instead of the normal ones only as normally required in (4.48). As a consequence, for a given interface triangle τ (a triangle whose vertices are located in the overlapping area), the components of (1.22) are considered as interface unknowns. Now, the problem at hand consists of two tasks: first, we need to construct an appropriate coupling between the interface flux components and the corresponding nodal unknowns (i.e. the components of the vector of conservative variables W for each vertex of the interface triangle τ); second, we have to define an associated linearization of the flux components for the implicit time integration. Using eq. (1.22) we can write:

$$\Upsilon_{\tau,i}(\tau) = \text{area}\,(\tau) \left[\begin{pmatrix} 0 \\ r_2(\tau) \\ r_3(\tau) \\ r_4(\tau) \end{pmatrix} \frac{\partial \varphi_i^\tau}{\partial x} + \begin{pmatrix} 0 \\ r_3(\tau) \\ s_3(\tau) \\ s_4(\tau) \end{pmatrix} \frac{\partial \varphi_i^\tau}{\partial y} \right] \tag{4.50}$$

where $r_i(\tau)$ and $s_i(\tau)$ are the components of the viscous flux vectors $\vec{\mathbb{F}}^v_x(\tau)$ and $\vec{\mathbb{F}}^v_y(\tau)$ which are constant on the triangle τ. According to the expressions of the components of $\vec{\mathbb{F}}^v_x(W)$ and $\vec{\mathbb{F}}^v_y(W)$, we can further write:

$$\Upsilon_{\tau,i}(\tau) = \text{area}\,(\tau) \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \partial_x \varphi_i^\tau & \partial_y \varphi_i^\tau & 0 & 0 & 0 \\ 0 & \partial_x \varphi_i^\tau & 0 & \partial_y \varphi_i^\tau & 0 \\ 0 & 0 & \partial_x \varphi_i^\tau & 0 & \partial_y \varphi_i^\tau \end{pmatrix} \begin{pmatrix} r_2(\tau) \\ r_3(\tau) \\ r_4(\tau) \\ s_3(\tau) \\ s_4(\tau) \end{pmatrix} \tag{4.51}$$

where $\partial_{x,y}\varphi_i^\tau = \dfrac{\partial \varphi_i^\tau}{\partial x, \partial y}$. The above formula can be seen as an alternative linearization of the viscous flux at the interface (with respect to the original one) where the new variables are given by the quantities $r_i(\tau)$ and $s_i(\tau)$. Therefore the coupling between the nodal variables located in the overlapping area will be replaced by the coupling between these variables and the new flux variables, this interaction being expressed via the 4×5 blocks of the formula (4.51).

Conversely, the coupling between the new interface variables and the nodal variables can be written using the classical linearization:

$$r_{2,3,4}(\tau) = \sum_{k_i \in \tau} \frac{\partial r_{2,3,4}(\tau)}{\partial W_{k_i}} W_{k_i} \quad \text{and} \quad s_{3,4}(\tau) = \sum_{k_i \in \tau} \frac{\partial s_{3,4}(\tau)}{\partial W_{k_i}} W_{k_i} \tag{4.52}$$

4.2.2.3 Formulation of an interface system

Putting together eq. (4.43) and (4.44) corresponding to the linearizations of internal and interface convective fluxes and, eq. (1.88)-(1.89) and (4.51)-(4.52) corresponding to the linearizations of internal and interface diffusive fluxes, we can form a global linear system that characterizes the discrete variant of additive Schwarz algorithm:

$$\begin{pmatrix} \mathcal{M}_1 & 0 & \mathcal{M}_{1c} & \mathcal{M}_{1v} \\ 0 & \mathcal{M}_2 & \mathcal{M}_{2c} & \mathcal{M}_{2v} \\ \mathcal{F}_{1c} & \mathcal{F}_{2c} & \mathsf{Id} & 0 \\ \mathcal{F}_{1v} & \mathcal{F}_{2v} & 0 & \mathsf{Id} \end{pmatrix} \begin{pmatrix} \delta W_1^{n+1} \\ \delta W_2^{n+1} \\ \delta \Phi_{\Gamma,c} \\ \delta \Phi_{\Gamma,v} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ 0 \\ 0 \end{pmatrix} \tag{4.53}$$

where $\mathcal{M}_1$ and $\mathcal{M}_2$ are the matrices resulting from the original linearization for vertices internal to Ω_1 and Ω_2. On the other hand, $\mathcal{F}_{1c}$, $\mathcal{F}_{2c}$, $\mathcal{F}_{1v}$, $\mathcal{F}_{2v}$, $\mathcal{M}_{1c}$, $\mathcal{M}_{1v}$, $\mathcal{M}_{2c}$ and $\mathcal{M}_{2v}$ are coupling matrices between internal and interface unknowns. Moreover, $\Phi_{\Gamma,c}$ is the vector of convective interface unknowns (see section 4.1.4.1.2) while $\Phi_{\Gamma,v}$ is the vector of diffusive interface unknowns with $\Phi_{\Gamma,v}(\tau) = (r_2(\tau)\ ,\ r_3(\tau)\ ,\ r_4(\tau)\ ,\ s_3(\tau)\ ,\ s_4(\tau))^T$. At this point, the internal unknowns can be eliminated in favor of the interface ones to yield the following interface system:

$$S \begin{pmatrix} \delta \Phi_{\Gamma,c} \\ \delta \Phi_{\Gamma,v} \end{pmatrix} = \begin{pmatrix} (\mathsf{Id} - S_{1c})\, \delta \Phi_{\Gamma,c} - S_{1v} \delta \Phi_{\Gamma,v} \\ -S_{2c} \delta \Phi_{\Gamma,c} + (\mathsf{Id} - S_{2v})\, \delta \Phi_{\Gamma,v} \end{pmatrix} = g \tag{4.54}$$

with:

$$\begin{cases} S_{1c} = \mathcal{F}_{1c}\mathcal{M}_1^{-1}\mathcal{M}_{1c} + \mathcal{F}_{2c}\mathcal{M}_2^{-1}\mathcal{M}_{2c} \ \text{ and } \ S_{1v} = \mathcal{F}_{1c}\mathcal{M}_1^{-1}\mathcal{M}_{1v} + \mathcal{F}_{2c}\mathcal{M}_2^{-1}\mathcal{M}_{2v} \\ S_{2c} = \mathcal{F}_{1v}\mathcal{M}_1^{-1}\mathcal{M}_{1c} + \mathcal{F}_{2v}\mathcal{M}_2^{-1}\mathcal{M}_{2c} \ \text{ and } \ S_{2v} = \mathcal{F}_{1v}\mathcal{M}_1^{-1}\mathcal{M}_{1v} + \mathcal{F}_{2v}\mathcal{M}_2^{-1}\mathcal{M}_{2v} \\ g = -\begin{pmatrix} [\mathcal{F}_{1c}\mathcal{M}_1^{-1} b_1 + \mathcal{F}_{2c}\mathcal{M}_2^{-1} b_2] \\ [\mathcal{F}_{1v}\mathcal{M}_1^{-1} b_1 + \mathcal{F}_{2v}\mathcal{M}_2^{-1} b_2] \end{pmatrix} \end{cases}$$

As usual in this context, once this system has been solved for $(\delta\Phi_c\ ,\ \delta\Phi_v)^T$, we obtain the values of the purely internal unknowns by performing independent (i.e. parallel) local solutions:

$$\begin{cases} \delta W_1^{n+1} = \mathcal{M}_1^{-1}(b_1 - \mathcal{M}_{1c}\delta\Phi_{\Gamma,c} - \mathcal{M}_{1v}\delta\Phi_{\Gamma,v}) \\ \delta W_2^{n+1} = \mathcal{M}_2^{-1}(b_2 - \mathcal{M}_{2c}\delta\Phi_{\Gamma,c} - \mathcal{M}_{2v}\delta\Phi_{\Gamma,v}) \end{cases} \tag{4.55}$$

Similarly to what has been done in section 4.1.4 for the solution of the Euler equations, the interface system (4.54) is solved using a full GMRES method while the multigrid by volume agglomeration method introduced in section 3.1 of chapter 3 is used to solved approximately the local systems (4.55).

4.2.3 Calculation of steady flows

In this section we apply the domain decomposition algorithm proposed in section 4.2.2 to the calculation of steady viscous flows and make some comparisons with the global approach described in section 1.4.1 of chapter 1.

4.2.3.1 Test cases definition

The test cases under consideration correspond to the calculation of external flows around a NACA0012 airfoil (see figure 4.6). In addition to mesh N3 of table 4.3, a finer discretization has been considered: mesh N4 contains 194,480 vertices, 387,584 elements and 582,064 edges. The following situations have been considered :

S1 : the transonic flow characterized by a free-stream Mach number $M_\infty = 0.85$ and an angle of attack $\theta = 0°$. The Reynolds number based on the airfoil chord length is equal to Re = 2000. The time step is obtained using the rule CFL=$500 \times i_t$ where i_t denotes the time step number.

S2 : the subsonic flow characterized by $M_\infty = 0.8$, $\theta = 10°$ and Re = 73. The time step is obtained using the rule CFL=$500 \times i_t$.

S3 : the supersonic flow characterized by $M_\infty = 2.0$, $\theta = 10°$ and Re = 106. The time step is obtained using the rule CFL=$50 \times i_t$.

The above test cases have been taken from a GAMM workshop dedicated to the solution of the two-dimensional Navier-Stokes equations for laminar viscous flows (see Bristeau *et al.*[88]). In each case, the calculation starts from a uniform flow characterized by $\rho_0 = 1$, $u_0 = 1$, $v_0 = 0$ and $p_0 = \dfrac{1}{\gamma_e M_\infty^2}$. Similarly to what has been done for the calculation of steady inviscid flows (see section 4.1.4.3), the results presented below are all characterized by the fact that the local systems induced by the domain decomposition solver are solved approximately.

4.2.3.2 Computing platforms and conventions

Numerical experiments have been performed on a cluster of 14 dual node PCs (each PC is equipped with two Pentium P3/500 Mhz and 512 Mb of SDRAM memory) running the Linux system and interconnected via an Ethernet 100 Mbit/s switch. The MPI implementation is MPICH. The code is written in Fortran 77 and the GNU G77 compiler has been used with maximal optimization options. Performance results are given for 64 bit arithmetic computations. In the following tables, N_p is the number of processes for the parallel execution (N_p also represents the number of subdomains); N_g is the total the number of grid levels (including the finest grid level) used in the multigrid solver for local systems; "# i_t" is the required number of pseudo-time steps to reach the steady state (convergence to the steady state is monitored using the normalized energy residual); "Total time" denotes the total simulation time and "CPU time" denotes the total CPU time (taken as the maximum value of the local per process measures); "% CPU" denotes the ratio of "CPU time" to "Total time". The parallel speedup $S(N_p)$ is always calculated using the elapsed execution times. Finally, the term "linear threshold" is used to characterize the accuracy of the linear system solves (i.e. the level of reduction of the initial linear residual).

4.2.3.3 S1 test case

Steady contour lines of the Mach number for this test case are shown on figure 4.15 (top figure). Performance results are given in tables 4.8 and 4.9 for calculations that have been performed using meshes N3 and N4. The first part of tables 4.8 and 4.9 is dedicated to global single grid computations. To be more precise, the original solver is adopted (see section 1.4.1 of chapter 1) and the global implicit system of eq. (1.84) is solved approximately using a Jacobi relaxation method (the corresponding strategy is denoted **GLOB.a** in the sequel). For this strategy, we have observed that imposing a linear threshold $\varepsilon_g = 10^{-1}$ at each time step results in the optimal non-linear convergence to steady state (in other words, reducing ε_g to 10^{-2} or below did not result in a reduction of the total simulation time). The second and third parts of tables 4.8 and 4.9 correspond to the application of the domain decomposition method developed in the present study. Two strategies have been considered; they differ from the complexity of the local solves: in the first strategy (denoted by **DDM/MG.a**) we impose the (local) linear threshold to $\varepsilon_l = 10^{-1}$ while the second strategy (denoted by **DDM/MG.b**) makes use of a constant complexity of 3 V(2,2,2)-cycles for each local solve. For both cases, the linear threshold for the interface system solver (a full GMRES method) is set to $\varepsilon_i = 10^{-1}$ and the V(2,2,2)-cycle is characterized by 2 pre- and 2 post-smoothing steps (the multigrid cycle is denoted by V(2,2,2)-cycle). In table 4.8, the parallel speedup is computed relatively to the

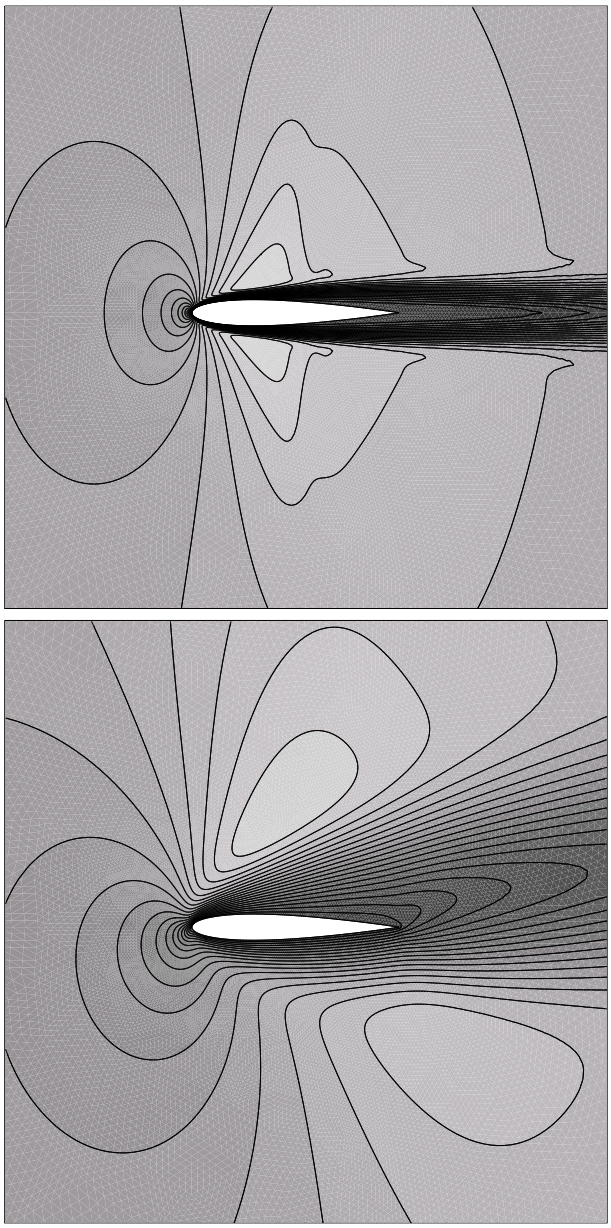


Figure 4.15: Viscous flow around a `NACA0012` airfoil
Steady Mach lines for the S1 (top) and S2 (bottom) test cases

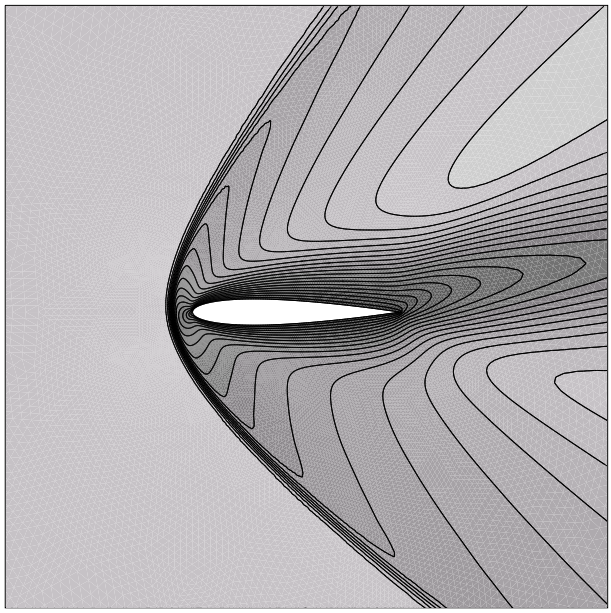


Figure 4.16: Viscous flow around a NACA0012 airfoil
Steady Mach lines for the S3 test case

elapsed time obtained for $N_p = 4$. The non-linear convergence to steady-state for the three solution strategies applied to mesh N3 and for $N_p = 24$ are shown on figure 4.17 (top figure). Finally, table 4.9 summarizes results of calculations performed using mesh N4 and $N_p = 16$.

This first series of results calls for several comments:

- as expected, the domain decomposition solver outperforms the global implicit solver for large numbers of subdomains only. For instance, for $N_p = 24$, the elapsed time for the global solution strategy based on the Jacobi solver is equal to 429 sec while the domain decomposition solver based on full GMRES for the interface system and a constant number of V(2,2,2)-cycles for the local solutions, yields an elapsed time of 239 sec. In this case, a 44% reduction in the total simulation time is obtained;

- the most remarkable characteristic of the domain decomposition solvers is their parallel efficiency which is here assessed by the "% CPU" ratio. This quantity degrades significantly for the global implicit solver when the number of subdomains is increased while it remains relatively high for the domain decomposition solver. In the conditions of the previous comparison, the improvement on the "% CPU" ratio is equal to 16%;

- the measures obtained for mesh N4 show that the domain decomposition approach based on a fixed linear threshold for the local solutions is much more costly than the global solution strategy. On the contrary, when a constant number of V(2,2,2)-cycles is used for the local solutions, a reduction of only 5% is obtained (while the corresponding measures for mesh N3 demonstrate a reduction of 27%). Higher gains are expected for larger numbers of subdomains since the cost of the local solutions is so far dominating the overall simulation time. Despite this fact, one can note that the parallel efficiency is approaching 98% for the domain decomposition solver while it is not higher than 90% for the global solver;

- in this study, the full GMRES iteration applied to the solution of the interface system (4.54) does not make use of any preconditioning technique. In this context, one may ask if this choice affects the scalability properties of the domain decomposition solver. A partial answer is given here on figure 4.17 (bottom figure) that shows the linear convergence of the domain decomposition solver based on a constant complexity of 3 V(2,2,2)-cycles for each local system solves. It is seen that the number of GMRES iterations slightly increases

when switching from $N_p = 4$ to $N_p = 24$ subdomains. However, this behavior has to be correlated with the fact that, in the present case, the interface systems are solved with a low accuracy therefore requiring only a few GMRES iterations (6 in average).

Table 4.8: S1 test case - timings for the steady state calculation (mesh N3)
Global solution strategy (parallel Jacobi linear solver) versus DDM/MG strategy (full GMRES)

Method	N_p	N_g	# i_t	CPU time	Total time	% CPU	$S(N_p)$
GLOB.a	4	1	73	1527 sec	1611 sec	95.0	1.0
	6	1	73	976 sec	1081 sec	90.5	1.5
	8	1	73	755 sec	856 sec	88.0	1.9
	12	1	73	495 sec	615 sec	80.5	2.6
	16	1	73	384 sec	512 sec	75.0	3.1
	24	1	73	272 sec	429 sec	63.5	3.8
DDM/MG.a	4	5	71	1679 sec	1751 sec	96.0	1.0
	6	5	72	1030 sec	1071 sec	96.0	1.6
	8	5	73	869 sec	926 sec	94.0	1.9
	12	5	74	599 sec	648 sec	92.5	2.7
	16	5	76	446 sec	493 sec	90.5	3.5
	24	4	78	286 sec	333 sec	86.0	5.3
DDM/MG.b	4	5	74	1296 sec	1320 sec	98.0	1.0
	6	5	75	844 sec	871 sec	97.0	1.5
	8	5	75	663 sec	681 sec	97.0	1.9
	12	5	79	489 sec	517 sec	94.5	2.5
	16	5	78	338 sec	372 sec	91.0	3.5
	24	4	77	212 sec	239 sec	89.0	5.5

Table 4.9: S1 test case - timings for the steady state calculation (mesh N4)
Global solution strategy (parallel Jacobi linear solver) versus DDM/MG strategy (full GMRES)

Method	N_p	N_g	# i_t	CPU time	Total time	% CPU
GLOB.a	16	1	151	3705 sec	4127 sec	90.0
DDM/MG.a	16	5	149	8338 sec	8535 sec	97.4
DDM/MG.b	16	5	161	3841 sec	3928 sec	98.0

4.2.3.4 S2 test case

Steady contour lines of the Mach number for this test case are shown on figure 4.15 (bottom figure). Performance results are given in tables 4.10 and 4.11 for calculations that have been performed using meshes N3 and N4. The selected solution strategies are basically the same than those considered for the S1 test case, except that for the domain decomposition solver, we have only applied a solution strategy based on a fixed number of V(2,2,2)-cycles for the local solutions (strategy **DDM/MG.b**). Concerning the calculations that have been performed using mesh N3, the remarks made for the S1 test case are still valid. Note that the domain decomposition solver always results in a lower number of pseudo-time steps to reach the steady state, except for $N_p = 12$. In that case we suspect that the partitioning of the global mesh has resulted in badly shaped subdomains with a direct impact on the quality of the produced coarse meshes via the agglomeration principle. Then, a local solution strategy

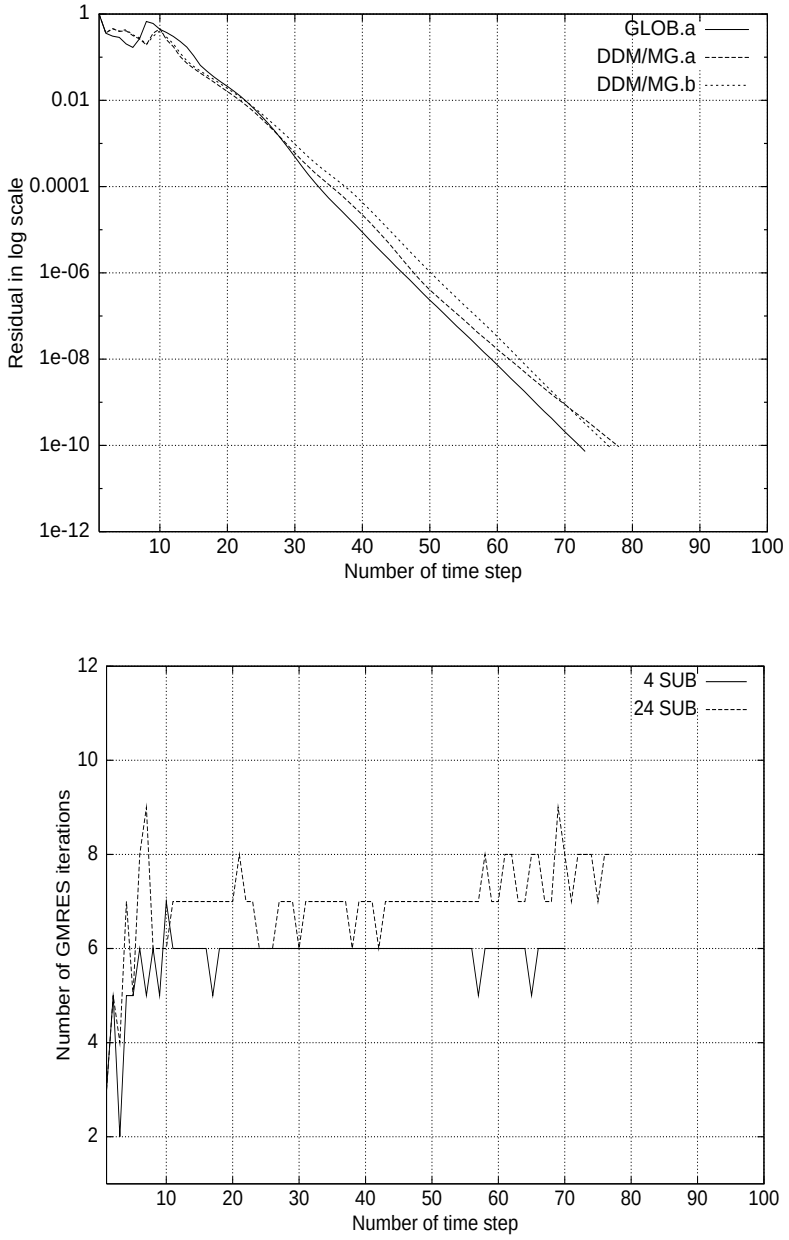


Figure 4.17: S1 test case - calculations with mesh N3

Top figure : non-linear convergence of the global and DDM/MG solvers
Bottom figure : linear convergence of the DDM/MG solver
for 4 and 24 subdomain decompositions
($\varepsilon_i = 10^{-1}$/local solutions: 3 V(2,2,2)-cycles)

based on more than 3 V(2,2,2)-cycles is probably necessary to recover a convergence to the steady state in a lower number of time iterations. The calculations based on mesh N4 show that the domain decomposition solver is at least 3.3 times faster than the global solution strategy. For the latter, reducing the linear threshold from $\varepsilon_g = 10^{-1}$ (strategy **GLOB.a**) to $\varepsilon_g = 10^{-2}$ (strategy **GLOB.b**) allows a faster convergence to steady state at the expense of a 5% increase in the total simulation time.

Table 4.10: S2 test case - timings for the steady state calculation (mesh N1)
Global solution strategy (parallel Jacobi linear solver) versus DDM/MG strategy (full GMRES)

Method	N_p	N_g	# i_t	CPU time	Total time	% CPU	$S(N_p)$
GLOB.a	4	1	90	1805 sec	1892 sec	95.5	1.0
	6	1	90	1164 sec	1270 sec	91.5	1.5
	8	1	90	876 sec	993 sec	88.5	1.9
	12	1	90	598 sec	723 sec	83.0	2.6
	16	1	90	450 sec	594 sec	76.0	3.2
	24	1	90	322 sec	488 sec	66.0	3.9
DDM/MG.b	4	5	84	1287 sec	1311 sec	98.0	1.0
	6	5	86	879 sec	896 sec	98.0	1.5
	8	5	84	641 sec	658 sec	97.5	2.0
	12	5	94	528 sec	552 sec	95.5	2.4
	16	5	84	313 sec	331 sec	94.5	4.0
	24	4	85	235 sec	261 sec	90.0	5.0

Table 4.11: S2 test case - timings for the steady state calculation (mesh N2)
Global solution strategy (parallel Jacobi linear solver) versus DDM/MG strategy (full GMRES)

Method	N_p	N_g	# i_t	CPU time	Total time	% CPU
GLOB.a	16	1	125	6211 sec	6903 sec	90.0
GLOB.b	16	1	117	6527 sec	7255 sec	90.0
DDM/MG.b	16	5	116	2043 sec	2090 sec	97.5

4.2.3.5 S3 test case

Steady contour lines of the Mach number for this test case are shown on figure 4.16. Performance results are given in tables 4.12 and 4.13 for calculations that have been performed using meshes N3 and N4. The first and the second parts of tables 4.12 and 4.13 are dedicated to computations based on the global implicit solver. Here, results are reported for two values of the linear threshold, $\varepsilon_g = 10^{-1}$ (strategy **GLOB.a**) and $\varepsilon_g = 10^{-2}$ (strategy **GLOB.b**). As will be seen below, the second value is the one that results in the faster non-linear convergence to steady state at the expense of total simulation times slightly higher than those obtained for the first value. The third and fourth parts of tables 4.12 and 4.13 correspond to the application of the domain decomposition solver. The strategies **DDM/MG.a** and **DDM/MG.b** introduced for the S1 test case have again been considered here. Table 4.13 gives a set of results for calculations performed using mesh N4 and $N_p = 16$. In table 4.8 the parallel speedup is computed relatively to the elapsed times obtained for $N_p = 4$. The non-linear convergence to steady-state for the four solution strategies and for $N_p = 24$ are presented on figure 4.18. In this figure, it is clear that the global solution strategy with $\varepsilon_g = 10^{-1}$, does not yield the correct convergence to steady state as compared to the one exhibited by the domain decomposition solver. The domain decomposition solver based on a constant number of V(2,2,2)-cycles for the local solutions is always yielding the shortest total simulation times.

For instance, for the calculations based on mesh N3, a comparison between the global solution strategy relying on the value $\varepsilon_g = 10^{-2}$ for the linear threshold and the domain decomposition strategy that uses a constant number of V(2,2,2)-cycles for the local system solves, shows a 66% reduction of the total simulation time for $N_p = 24$ (470 sec for the global solver and 161 sec for the domain decomposition solver). A similar comparison for calculations based on mesh N4 and $N_p = 16$ shows that the domain decomposition solver is 4.2 times faster than the global solution strategy. For the latter, reducing the linear threshold from $\varepsilon_g = 10^{-1}$ to $\varepsilon_g = 10^{-2}$ allows a notable reduction in the number of time steps required to reach the steady state at the expense of a 36% increase in the total simulation time.

Table 4.12: S3 test case - timings for the steady state calculation (mesh N1)
Global solution strategy (parallel Jacobi linear solver) versus DDM/MG strategy (full GMRES)

Method	N_p	N_g	# i_t	CPU time	Total time	% CPU	$S(N_p)$
GLOB.a	4	1	155	1399 sec	1463 sec	95.5	1.0
	6	1	155	916 sec	999 sec	91.5	1.5
	8	1	155	670 sec	754 sec	89.0	2.0
	12	1	155	454 sec	552 sec	82.5	2.6
	16	1	155	323 sec	434 sec	74.0	3.4
	24	1	155	244 sec	367 sec	66.5	4.0
GLOB.b	4	1	66	1690 sec	1769 sec	98.0	1.0
	6	1	66	1103 sec	1212 sec	91.0	1.5
	8	1	66	804 sec	937 sec	86.0	1.9
	12	1	66	555 sec	689 sec	80.5	2.5
	16	1	66	413 sec	552 sec	74.5	3.2
	24	1	66	295 sec	470 sec	63.0	3.8
DDM/MG.a	4	5	57	1163 sec	1202 sec	97.0	1.0
	6	5	58	764 sec	790 sec	96.5	1.5
	8	5	57	554 sec	586 sec	94.5	2.0
	12	5	63	419 sec	455 sec	92.0	2.6
	16	5	57	251 sec	284 sec	88.0	4.2
	24	4	58	177 sec	206 sec	86.0	5.8
DDM/MG.b	4	5	57	967 sec	979 sec	99.0	1.0
	6	5	58	646 sec	661 sec	98.0	1.5
	8	5	57	454 sec	472 sec	96.0	2.1
	12	5	62	350 sec	369 sec	95.0	2.7
	16	5	57	225 sec	243 sec	92.5	4.0
	24	4	58	145 sec	161 sec	90.0	6.1

4.2.4 Calculation of unsteady flows

In this section we apply the domain decomposition algorithm proposed in section 4.2.2 to the calculation of unsteady viscous flows. For this purpose, we consider the global implicit approach described in section 1.4.2 of chapter 1. It is clear that the method proposed in section 4.2.2 can be applied to each of the two steps of eq. (1.94) characterizing the second order linearized implicit scheme under consideration. Indeed, at each step of eq. (1.94), an interface system (4.54) can be formed and subsequently solved using a full GMRES method. In the following, we consider and compare the performances of two parallel solution strategies:

- a global multigrid approach: the parallelized multigrid by volume agglomeration method (see section 3.1 of chapter 3) is applied to the solution of the two linear systems resulting from eq. (1.94). In that case, the Jacobi relaxation method is playing the role of the smoother in a multigrid V-cycle;

Table 4.13: S3 test case - timings for the steady state calculation (mesh N2)
Global solution strategy (parallel Jacobi linear solver) versus DDM/MG strategy (full GMRES)

Method	N_p	N_g	# i_t	CPU time	Total time	% CPU
GLOB.a	16	1	208	4693 sec	5242 sec	89.5
GLOB.b	16	1	93	6420 sec	7152 sec	90.0
DDM/MG.a	16	5	93	2206 sec	2475 sec	89.0
DDM/MG.b	16	5	91	1677 sec	1715 sec	98.0

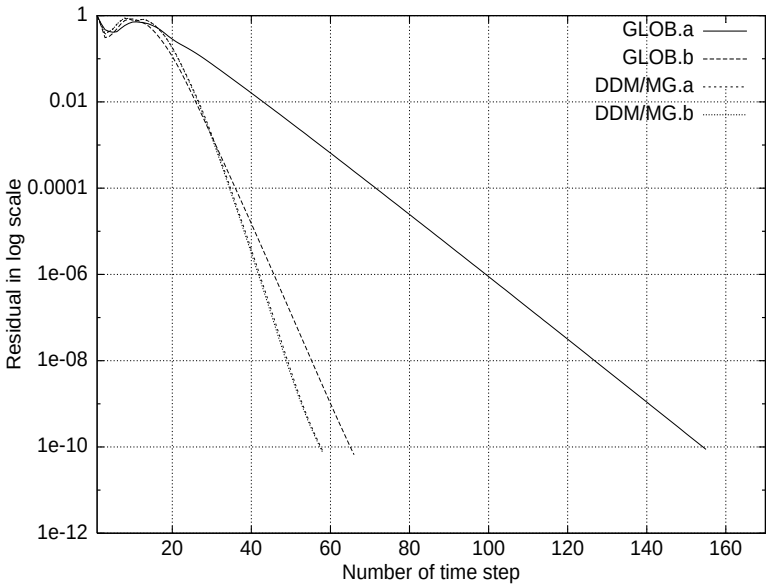


Figure 4.18: S3 test case - calculations with mesh N1
Non-linear convergence of the global and DDM/MG solvers

- an hybrid domain decomposition/multigrid method: the algorithm proposed in section 4.2.2 is used to solve the interface systems resulting from the two steps of eq. (1.94). In that case, the multigrid by agglomeration domain is used to solve the linear system obtained at each iteration of the GMRES method applied to the solution of a given interface system. A point-wise Gauss-Seidel relaxation method is playing the role of the smoother in a multigrid V-cycle.

4.2.4.1 Test case definition

The test case selected here consists in the numerical simulation of an unsteady flow around a `NACA0012` airfoil. The simulation starts from a uniform flow configuration characterized by:

- $\rho_0 = 1$,
- $\rho_0 u_0 = \cos(\theta)$ and $\rho_0 v_0 = \sin(\theta)$ with $\theta = 20°$,
- $E_0 = \frac{1}{2}\rho_0 \left(u_0^2 + v_0^2\right) + \frac{p_0}{\gamma_e - 1}$ with $p_0 = \frac{\rho_0 \left(u_0^2 + v_0^2\right)}{\gamma_e M_\infty^2}$.

The flow is assumed subsonic (the free-stream Mach number is equal to $M_\infty = 0.2$) and laminar (Reynolds number has been set to 2100). A global time step strategy is selected. In these conditions, after a transitory phase, the flow exhibits a permanent regime characterized by a periodic shedding of vertices at the trailing edge of the airfoil. The flow snapshots shown on figures 4.19 and 4.20 correspond to a particular time interval such that the periodic regime is well established. The computational mesh contains 194,480 vertices, 387,584 elements and 582,064 edges. The corresponding calculation has been performed for a constant CFL equal to 100 and has required 1175 time steps. Note however that the execution times reported in the following tables correspond to 100 time steps.

4.2.4.2 Computing platforms and notational conventions

Numerical experiments have been performed on several clusters of PCs:

- a cluster consisting of 14 dual nodes Pentium III/500 Mhz and 19 dual nodes Pentium III/933 Mhz, with 512 Mb of SDRAM 133 Mhz each, interconnected by an Ethernet 100 Mbit/s switch.
- a cluster of 16 dual nodes Pentium IV/2 Ghz with 1 Gb of RDRAM 800 Mhz each, interconnected by an Ethernet 1 Gbit/s switch.
- a cluster consisting of 216 HP e-vectra nodes (Pentium III/733 Mhz with 256 Mb of SDRAM 133 Mhz each) interconnected by several Ethernet 100 Mbit/s switches.
- a cluster of 32 dual nodes Pentium III/1 Ghz with 512 Mb of SDRAM 133 Mhz each, interconnected by a Myrinet 2 Gbit/s network.

All these clusters are running the Linux operating system. Performance results are given for 64 bit arithmetic computations. The code is written in Fortran 77 and parallel programming relies on the MPICH implementation of MPI. In the following tables and discussions, N_p is the number of processes for the parallel execution (N_p also represents the number of subdomains), N_n is the number of processing nodes (since most of the clusters described previously are based on dual boards, this figure will allow to make the distinction between calculations using one or two processes on a given node in order to assess the impact of the local memory architecture), N_g is the total number of grid levels (including the finest grid level) used in a multigrid solution strategy. Moreover, "Total time" denotes the total (wall clock) simulation time and "CPU time" stands for the corresponding total CPU time taken as the maximum of the per process values. Finally, "%CPU" is the ratio of the total CPU time to the total wall clock time. The parallel speedup $S(N_p)$ is always calculated using the total wall clock times.

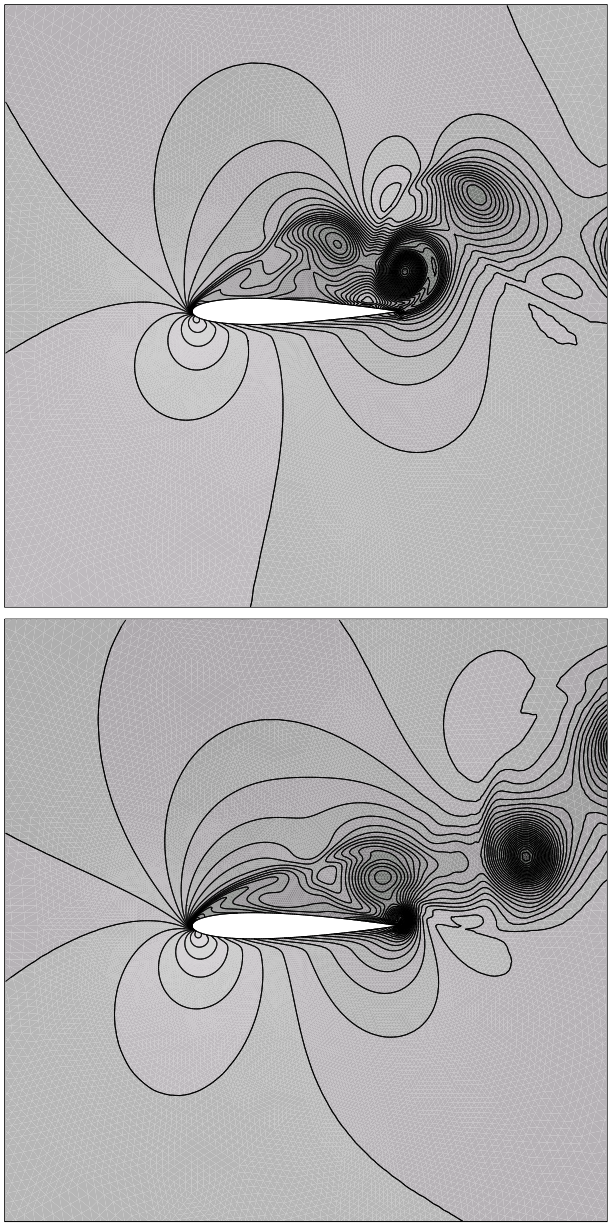


Figure 4.19: Unsteady flow around a `NACA0012` airfoil
Unsteady density contour lines at various times

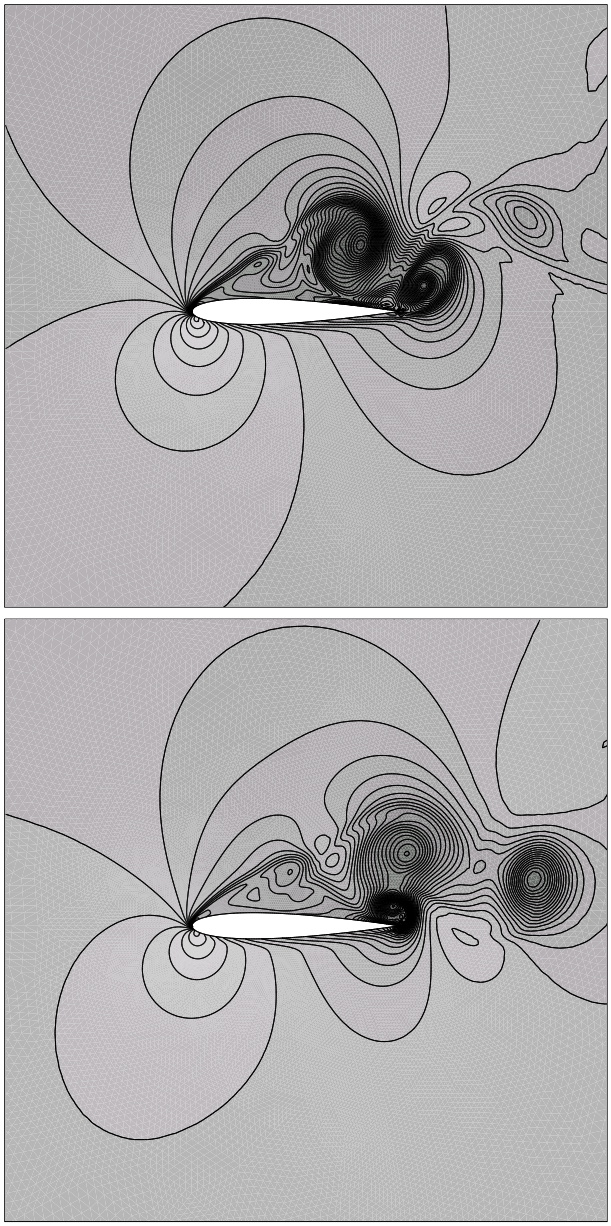


Figure 4.20: Unsteady flow around a `NACA0012` airfoil
Unsteady density contour lines at various times

4.2.4.3 Solution methods

Here, the linear threshold ε is used to characterize the accuracy of the linear system solves. More precisely, ε stands for the level of reduction of the normalized linear residual. The solution methods differ through the strategy used for solving the linear systems obtained at each time step:

- **MG** : the parallelized multigrid method. A V(2,2,2) cycle has been selected (i.e, using 2 pre- and post-smoothing iterations and 2 iterations for the solution of the problem on the coarsest grid level). The linear threshold has been set to $\varepsilon_{\text{mg}} = 10^{-1}$.

- **DDM/MG.a** : the hybrid domain decomposition/multigrid method where the GMRES method is used to solve the interface problem with a linear threshold that has been set to $\varepsilon_{\text{ddm}} = 10^{-1}$. The local linear systems obtained at each GMRES iteration are (approximately) solved using several V(2,2,2) multigrid cycles with a (local) linear threshold that has been set to $\varepsilon_{\text{mg}} = 10^{-1}$.

- **DDM/MG.b** : the **DDM/MG.a** strategy where the local systems are solved using only 1 V(2,2,2) multigrid cycle, all other components and parameters being unchanged.

4.2.4.4 Assessment of solution accuracy

Given the above choices for solving the linear systems resulting form the implicit time integration scheme of section 1.4.2 of chapter 1, one important question is concerned with the overall accuracy of the computed unsteady flows. A partial answer to this question is given on fig. 4.21 that depicts the evolution of the lift coefficient as a function of the physical time for the numerical simulations corresponding to the selected solution methods. One important point to note is that, despite its minimal constant complexity for the local solves, the **DDM/MG.b** solver results in a physical solution which is very similar to the ones resulting from the application of the **MG** and **DDM/MG.b** solvers.

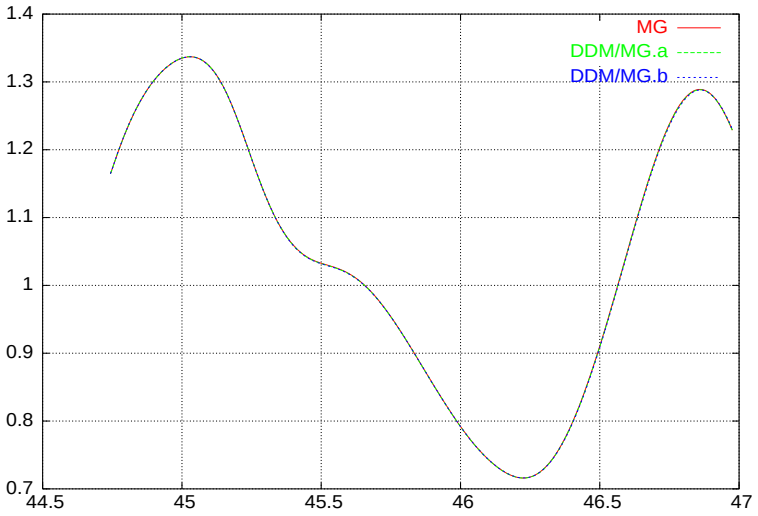


Figure 4.21: Unsteady flow around a `NACA0012` airfoil
Evolution of the lift coefficient (Y-axis) versus time (X-axis)

4.2.4.5 Performance results

Performance results are summarized in tables 4.14 to 4.19. Note that for some clusters results are given for both the GNU G77 and Portland Group PGF77 compilers. These results call for several comments:

- the most remarkable characteristic of the proposed hybrid domain decomposition/multigrid method certainly is its parallel efficiency. Indeed, the "%CPU" ratio degrades significantly for the multigrid ($\mathbf{MG}$) solver when the number of subdomains is increased while it remains relatively high for the hybrid domain decomposition/multigrid solvers ($\mathbf{DDM/MG.a}$ and $\mathbf{DDM/MG.b}$). The higher computational efficiency of the $\mathbf{DDM/MG.a}$ solver is essentially due to a better computation/communication ratio since, in most cases, several V(2,2,2) multigrid cycles are required to reach the prescribed linear threshold (i.e $\varepsilon_{\text{mg}} = 10^{-1}$). Note that, for a given partition, the number of V(2,2,2) cycles may vary from one subdomain to another due to different local flow features. In contrast, the values of the parallel speedup are often better for the $\mathbf{DDM/MG.b}$ solver which is characterized by a constant complexity for the local solves. On the cluster of Pentium III/733 Mhz nodes with Ethernet 100 Mbit/s interconnection (table 4.19), the overall gain resulting from the hybrid domain decomposition/multigrid strategy reaches 27% for the $\mathbf{DDM/MG.a}$ solver and 13% for the $\mathbf{DDM/MG.b}$ when $N_p = 112$.

- The performance gap between the $\mathbf{MG}$ and $\mathbf{DDM/MG}$ solvers is particularly evident for the measures obtained with the PGF77 compiler. For instance, for the calculations performed on the cluster of Pentium III/933 Mhz nodes with Ethernet 100 Mbit/s interconnection (tables 4.15 and 4.16), the CPU time of the $\mathbf{MG}$ solver when $N_p = N_n = 14$ is equal to 523 sec for the G77 compiler and 360 sec for the PGF77 compiler. However, the corresponding wall clock times are respectively equal to 626 sec and 643 sec and the "%CPU" degrades from 83.6% to 56.0%. In comparison, the measures obtained for the $\mathbf{DDM/MG.b}$ solver are 655 sec/672 sec (CPU/REAL) for the G77 compiler and 626 sec/746 sec for the PGF77 compiler. In that case the CPU utilization degrades from 97.5% to 83.4%. It is worthwhile to note that the results obtained on the cluster of Pentium III/1 Ghz nodes with Myrinet 2 Gbit/s interconnection (see table 4.17) are consistently better and in favour of the $\mathbf{MG}$ solver even though the performance gap with the $\mathbf{DDM/MG.b}$ solver decreases as the number of processes increases.

- As a general rule, the parallel speedups are better for the calculations for which only one process is running on each computational node. This is easily explained by the fact that the numerical kernels considered here are essentially those characterizing sparse matrix computations with irregular and indirect addressing of data in memory. Moreover, the size of L2 caches is relatively small (256 Kb or 512 Kb depending on the processor type). Therefore, the characteristics of RAM memory and memory bus plays a major role in the achievable speedup on a given dual node. Comparing the CPU times obtained for $N_p = N_n = 14$ and $N_p = 14$ with $N_n = 7$ shows a performance degradation ranging from 24% to 26% on the cluster of Pentium III/1 Ghz nodes with Myrinet 2 Gbit/s interconnection and, from 14.5% to 16.5% on the cluster of Pentium IV/2 Ghz with Ethernet 1 Gbit/s interconnection. It is clear that the RDRAM of the Pentium IV based node has a notable impact on the achievable floating point performance for the sparse type operations characterizing the underlying flow solver.

Table 4.14: Cluster of Pentium III/500 Mhz (Ethernet 100 Mbit/s interconnection) G77 compiler using `-O3 -ffast-math -static` options

Method	N_p	N_n	N_g	CPU time	Total time	% CPU	$S(N_p)$
MG	7	4	5	1203 sec	1257 sec	95.7%	1.00
-	14	7	4	584 sec	629 sec	92.9%	1.99
-	28	14	4	323 sec	369 sec	87.6%	3.50
MG	7	7	5	973 sec	1012 sec	96.2%	1.00
-	14	14	4	488 sec	515 sec	94.8%	1.96
DDM/MG.a	7	4	5	2912 sec	2954 sec	98.6%	1.00
-	14	7	4	1825 sec	1865 sec	97.9%	1.58
-	28	14	4	933 sec	1013 sec	92.1%	2.92
DDM/MG.a	7	7	5	2171 sec	2246 sec	96.7%	1.00
-	14	14	4	1472 sec	1500 sec	98.2%	1.49
DDM/MG.b	7	4	5	1780 sec	1815 sec	98.1%	1.00
-	14	7	4	1010 sec	1042 sec	97.0%	1.74
-	28	14	4	500 sec	520 sec	96.2%	3.49
DDM/MG.b	7	7	5	1403 sec	1425 sec	98.5%	1.00
-	14	14	4	807 sec	827 sec	97.6%	1.72

Table 4.15: Cluster of Pentium III/933 Mhz (Ethernet 100 Mbit/s interconnection) G77 compiler using `-O3 -ffast-math -static` options

Method	N_p	N_n	N_g	CPU time	Total time	% CPU	$S(N_p)$
MG	7	4	5	1066 sec	1134 sec	94.0%	1.00
-	14	7	4	545 sec	696 sec	78.3%	1.63
-	28	14	4	301 sec	555 sec	54.3%	2.05
MG	7	7	5	1040 sec	1111 sec	93.7%	1.00
-	14	14	4	523 sec	626 sec	83.6%	1.77
DDM/MG.a	7	4	5	2824 sec	2902 sec	97.4%	1.00
-	14	7	4	1795 sec	1857 sec	96.7%	1.56
-	28	14	4	927 sec	1048 sec	88.5%	2.77
DDM/MG.a	7	7	5	2687 sec	2739 sec	98.1%	1.00
-	14	14	4	1660 sec	1690 sec	98.5%	1.62
DDM/MG.b	7	4	5	1664 sec	1696 sec	98.2%	1.00
-	14	7	4	1075 sec	1103 sec	97.5%	1.54
-	28	14	4	490 sec	577 sec	85.0%	2.94
DDM/MG.b	7	7	5	1134 sec	1154 sec	98.3%	1.00
-	14	14	4	655 sec	672 sec	97.5%	1.71

Table 4.16: Cluster of Pentium III/933 Mhz (Ethernet 100 Mbit/s interconnection) PGF77 compiler using `-pc 64 -fast -Mvect -Mdalign` options

Method	N_p	N_n	N_g	CPU time	Total time	% CPU	$S(N_p)$
MG	7	4	5	1077 sec	1279 sec	84.3%	1.00
-	14	7	4	563 sec	928 sec	60.7%	1.38
-	28	14	4	315 sec	1028 sec	30.6%	1.24
MG	7	7	5	697 sec	812 sec	85.8%	1.00
-	14	14	4	360 sec	643 sec	56.0%	1.26
DDM/MG.a	7	4	5	2863 sec	2959 sec	96.8%	1.00
-	14	7	4	1833 sec	1998 sec	91.8%	1.48
-	28	14	4	974 sec	1217 sec	80.0%	2.43
DDM/MG.a	7	7	5	1801 sec	1837 sec	98.0%	1.00
-	14	14	4	1203 sec	1327 sec	90.7%	1.38
DDM/MG.b	7	4	5	1665 sec	1740 sec	95.7%	1.00
-	14	7	4	1010 sec	1152 sec	87.7%	1.51
-	28	14	4	523 sec	724 sec	72.3%	2.40
DDM/MG.b	7	7	5	1200 sec	1258 sec	95.4%	1.00
-	14	14	4	626 sec	746 sec	83.4%	1.69

Table 4.17: Cluster of Pentium III/1 Ghz (Myrinet 2 Gbit/s interconnection) PGF77 compiler using `-pc 64 -fast -Mvect -Mdalign` options

Method	N_p	N_n	N_g	CPU time	Total time	% CPU	$S(N_p)$
MG	14	7	4	403 sec	404 sec	99.7%	1.00
-	28	14	4	222 sec	223 sec	99.5%	1.81
-	56	28	4	135 sec	136 sec	99.2%	2.97
MG	14	14	4	306 sec	307 sec	99.6%	1.00
-	28	28	4	166 sec	167 sec	99.4%	1.84
DDM/MG.a	14	7	4	1379 sec	1381 sec	99.8%	1.00
-	28	14	4	775 sec	776 sec	99.8%	1.78
-	56	28	4	385 sec	386 sec	99.7%	3.58
DDM/MG.a	14	14	4	1045 sec	1046 sec	99.9%	1.00
-	28	28	4	586 sec	587 sec	99.8%	1.78
DDM/MG.b	14	7	4	740 sec	742 sec	99.7%	1.00
-	28	14	4	365 sec	366 sec	99.7%	2.00
-	56	28	4	192 sec	193 sec	99.5%	3.84
DDM/MG.b	14	14	4	545 sec	546 sec	99.8%	1.00
-	28	28	4	274 sec	275 sec	99.6%	1.98

Table 4.18: Cluster of Pentium IV/2 Ghz (Ethernet 1 Gbit/s interconnection) G77 compiler using `-O3 -ffast-math -static` options

Method	N_p	N_n	N_g	CPU time	Total time	% CPU	$S(N_p)$
MG	7	4	5	403 sec	422 sec	95.5%	1.00
-	14	7	4	182 sec	197 sec	92.4%	2.14
-	28	14	4	93 sec	110 sec	84.6%	3.84
MG	7	7	5	345 sec	359 sec	96.1%	1.00
-	14	14	4	152 sec	165 sec	92.2%	2.17
DDM/MG.a	7	4	5	958 sec	980 sec	97.8%	1.00
-	14	7	4	617 sec	629 sec	98.1%	1.56
-	28	14	4	310 sec	334 sec	92.8%	2.93
DDM/MG.a	7	7	5	832 sec	853 sec	97.6%	1.00
-	14	14	4	529 sec	538 sec	98.4%	1.58
DDM/MG.b	7	4	5	605 sec	617 sec	98.1%	1.00
-	14	7	4	332 sec	343 sec	96.8%	1.80
-	28	14	4	155 sec	164 sec	94.6%	3.76
DDM/MG.b	7	7	5	521 sec	542 sec	96.2%	1.00
-	14	14	4	284 sec	292 sec	97.3%	1.85

Table 4.19: Cluster of Pentium III/733 Mhz (Ethernet 100 Mbit/s interconnection) G77 compiler using `-O3 -ffast-math -static` options

Method	N_p	N_n	N_g	CPU time	Total time	% CPU	$S(N_p)$
MG	28	28	4	253 sec	331 sec	76.5%	1.00
-	56	56	4	134 sec	210 sec	63.9%	1.58
-	112	112	3	98 sec	212 sec	46.3%	1.56
DDM/MG.a	28	28	4	810 sec	910 sec	89.0%	1.00
-	56	56	4	398 sec	463 sec	86.0%	1.97
-	112	112	3	209 sec	285 sec	73.5%	3.19
DDM/MG.b	28	28	4	391 sec	460 sec	85.0%	1.00
-	56	56	4	193 sec	255 sec	75.7%	1.80
-	112	112	3	103 sec	174 sec	59.2%	2.65

4.3 Conclusion

These last ten years, the design of domain decomposition methods for the solution of hyperbolic and mixed hyperbolic/parabolic systems of PDEs has been the subject of active researches. Without entering into all the details, one can reasonably state that such studies fall into one of the two following categories:

- development of domain decomposition solvers based on appropriate formulations at the continuous level, Quarteroni and Stolcis[116], Quarteroni and Valli[118] and Paraschivoiu[112].
- methods that exploit domain decomposition approach to construct a parallel (additive) preconditioner (Schwarz or Schur complement type formulations) to a linear or non-linear system of equations, see among others Barth *et al.*[9], Cai *et al.*[19]-[20]-[21], Tidriri[136], Wu *et al.*[146] and Sala[125].

In this chapter, we have described our contributions concerning the development and analysis of domain decomposition methods for the solution of the two-dimensional Euler and Navier-Stokes equations modeling compressible inviscid and laminar viscous flows. The common framework of the methods proposed here consists in the formulation of an additive Schwarz algorithm involving transmission conditions that are derived from the weak formulation of the underlying boundary value problem. In the case of inviscid flows, these interface conditions are equivalent to requiring the continuity of the components of the normal convective flux. For laminar viscous flows, the interface conditions generalize the one used for the Euler conditions by imposing in addition the continuity of the components of the normal viscous flux.

For the purely hyperbolic system of PDEs given by the Euler equations, we have studied the convergence of the proposed additive Schwarz algorithm. By considering the linearized equations and assuming a specific type of decomposition of the real space, it is possible to apply Fourier analysis to obtain an explicit expression of the convergence rate of the additive Schwarz algorithm. Both two- and three-dimensional equations have been treated as well as overlapping and non-overlapping domain decompositions. However, this study has been limited to two- and three-subdomain decompositions. Experimental results have been provided for the numerical solution of the two-dimensional Euler equations using an implementation of the additive Schwarz algorithm in the context of the finite volume formulation on triangular meshes described in section 1.2.1 of chpater 1. This implementation of the Schwarz algorithm is based on a non-overlapping element-based partitioning where the elements are the finite volume cells. In general, the experimental results are in qualitative agreement with those characterizing the convergence analysis of an overlapping Schwarz algorithm in the continuous case. Recently [42], we have conducted a similar convergence analysis at the discrete level assuming a finite volume formulation on a quadrilateral grid. This study has allowed us to further explain the behavior obtained experimentally.

The second part of this chapter has been devoted to the calculation of inviscid and laminar viscous flows in the context of the mixed element/volume formulation on triangular meshes described in section 1.2.1 of chapter 1. In the latter case, both steady and unsteady flows have been considered. Discrete counterparts of the domain decomposition solvers formulated in the continuous case have been developed in conjunction with first order and second order implicit linearized schemes for the time integration of the Euler and Navier-Stokes equations. A distinctive feature of our implementation is the use of a multigrid by agglomeration technique for the solution of the local problems obtained at each iteration of the interface system solver. By doing so, we obtain hybrid domain decomposition/multigrid methods for the solution of the sparse linear systems obtained at each implicit time step. Such a method can also be viewed as a particular form of parallel multigrid in which multigrid acceleration is applied to the iterative solution of the local linear systems induced by a Schwarz type domain decomposition algorithm. In this study, we have investigated numerically the effect of an approximate solution of local problems on the overall efficiency of the domain decomposition solver. In practice, this strategy does not affect the convergence of the domain decomposition solver and actually is mandatory to make the domain decomposition solver competitive with classical (global) monogrid or multigrid solution techniques. As a matter of fact, the results presented here have shown that, for a sufficiently large number of subdomains, the resulting domain decomposition solvers are more efficient than classical global solution strategies: first, higher parallel efficiencies are obtained because of reduced communication costs, especially when the number of subdomains is increased; second, the use of a multigrid principle for the local system solves largely contributes to improve the numerical efficiency of the domain decomposition solvers.

The performances of these hybrid domain decomposition/multigrid methods have been evaluated from the numerical and parallel efficiency viewpoints, using several clusters of PCs with different computational nodes and interconnection networks. For clusters of PCs based on a high performance network such as Myrinet, the use of a hybrid domain decomposition/multigrid ($\mathbf{DDM/MG}$) method is questionable since a global parallel multigrid ($\mathbf{MG}$) method scales reasonably well and simulations times are always in favor of this method. However, we note that, as the number of subdomains increases, the gap between the hybrid $\mathbf{DDM/MG}$ and global $\mathbf{MG}$ methods decreases which suggest that for larger numbers of subdomains than the maximum value considered here for this type of cluster, the hybrid $\mathbf{DDM/MG}$ method will outperform the global $\mathbf{MG}$ method. For the other types of clusters which are all based on Ethernet 100 Mbit/s or Ethernet 1 Gbit/s interconnections, performance results show that the hybrid $\mathbf{DDM/MG}$ method consistently demonstrate higher parallel efficiencies than the global $\mathbf{MG}$ method. In addition, when the complexity of the local solves is set to a fixed number of multigrid cycles, the resulting hybrid $\mathbf{DDM/MG}$ method becomes a viable alternative to the global $\mathbf{MG}$ method from the point of view of the total simulation time. It is expected that experimental results will be even more in favor of the hybrid $\mathbf{DDM/MG}$ method in the context of the solution of 3D flows which will be the focus of our future works.

Acknowledgements. The cluster of HP e-vectra (Pentium III/733 Mhz) nodes is located at IMAG in Grenoble. The cluster of Pentium III/1 Ghz nodes with Myrinet interconnection is located at the CEMEF laboratory of Ecole des Mines in Sophia Antipolis. The other clusters are located at the INRIA center in Sophia Antipolis.

Conclusions and perspectives

In this document, I have described several research projects in which I have been involved while I was a member of the Sinus team at INRIA Sophia Antipolis. This covers the period from November 1993 to October 2000. Some of these works are currently the subject of further studies and developments in various contexts.

As mentioned in section 3.1 of chapter 3, the developments concerning the linear multigrid by volume agglomeration method for the acceleration of three-dimensional flow calculations on unstructured meshes took place in the N3S-NATUR industrial CFD software. However, at the time of this study, the monogrid version of this software was still evolving to include complex physical models such as chemical reactions for combustion problems and additional numerical ingredients for turbomachinery and combustion engine applications. These physical and numerical features were not fully taken into account in our contributions that, from this point of view, have resulted in a pre-industrial multigrid version of N3S-NATUR. Thanks to the encouraging results demonstrated in this study for some typical industrial calculations, the multigrid version of N3S-NATUR is now considered for several upgrades that are conducted by Snecma and Simulog.

With regards to the non-linear multi-mesh multigrid algorithms discussed in section 3.3 of chapter 3, we first note that additional research and development activities have been conducted in order to extend the applicability of these algorithms to laminar viscous flows[27] and turbulent viscous flows[126]. In the case of turbulent viscous flows, a specific study was concerned with the assessment of the performances of the FAS-MG and FMG algorithms in conjunction with highly stretched meshes[17]. Finally, it is worthwhile to mention that the software kernels resulting from the IDeMAS are currently exploited by some of the industrial partners of the project: Daimler-Benz Aerospace (DASA) has adopted THOR as a component of their suite of CFD simulation codes whereas Dassault Aviation has integrated the multigrid oriented modules within their finite element unstructured mesh flow solver based on SUPG and MDHR schemes.

Domain decomposition methods for the solution of hyperbolic and mixed hyperbolic/parabolic systems of PDEs are still the subject of active studies worldwide. Our contributions in this field and more particularly, for the numerical resolution of the systems of Euler and Navier-Stokes equations modeling compressible flows, have been described in chapter 4 and are currently continuing to evolve in collaboration with V. Dolean (University of Evry and CMAP, Ecole Polytechnique, Palaiseau) and F. Nataf (CMAP). Two main objectives are pursued: (1) the design of domain decomposition methods for the calculation of three-dimensional flows and, (2) the construction of optimized interface conditions for the acceleration of the convergence of additive Schwarz algorithms in the context of inviscid flows. Clearly, the first point consists in the extension of the algorithms proposed in sections 4.1.4, 4.2.3 and 4.2.4 to the solution of the three-dimensional systems of Euler and Navier-Stokes equations. This will be done in the context of an exsisting parallel flow solver[84] which is based on the mixed finite element/volume approximation method on unstructured tetrahedral meshes discussed in section 1.2.1 of chapter 1. Moreover, both overlapping and non-overlapping additive Schwarz algorithms will be studied. For what concern the second point, the goal is to study optimized interface conditions for the system of Euler equations in the spirit of works previously done for a convection/diffusion problem[76]-[77]. Preliminary results in this direction are presented in [41].

In November 2000, I moved from the Sinus team to the Caiman team at INRIA Sophia Antipolis. The research activities undertaken in the Caiman team aim at designing, analyzing and implementing numerical methods for the simulation of wave propagation phenomena characterizing electromagnetic, elastodynamic and aeroacoustic problems. The numerical methods at the heart of these studies are directly inspired from those previsouly designed

for CFD applications. For instance, several contributions of the Caiman team are concerned with finite volume type approximations on unstructured meshes for the solution of the time domain Maxwell equations which is a hyperbolic system of PDEs. While the first works were conducted using upwind schemes[31], the focus is actually on centered finite volume[114] and discontinuous Galerkin[113] schemes. The stability of these schemes is demonstrated thanks to the conservation of a discrete form of the electromagnetic energy. These finite volume time domain (FVTD) schemes represent viable alternatives to the well know finite difference time domain (FDTD) method due to Yee[149] whose application is essentially limited to uniform or locally refined cartesiang grids. In this context, several new research topics could also benefit from some of the works that have been reported in this document. Two examples of such topics that will be part of my near future research activities are (1) the design of implicit time integration schemes for the time domain Maxwell equations in conjuction with the above cited FVTD methods and, (2) the construction of domain decomposition methods for the coupling of non-matching, regular (for the FDTD method) and irregular (for FVTD methods) meshes. Moreover, one emerging domain of application of computational electromagnetism is concerned with the study of the interaction between electromagnetic fields and living tissues for biological[2], environmental safety and therapeutic or diagnostic medical issues[122]. In addition to their societal implication, these applications define challenging problems for numerical methods such as those discussed here since they involve highly heterogenous media and irregular computational domains (e.g. human head tissues in the numerical dosimetry of radio-frequency fields issued from mobile phones). Finally, from the implementation viewpoint, the availability of parallel platforms consisting of clusters of multiprocessor nodes motivate the study of hybrid parallel paradigms mixing shared memory programming, through OpenMP directives, and message passing programming, via MPI. It is worthwhile to note that such architectures should also lead to the development of new parallel algorithms combining numerical kernels that are individually well suited to one of these two parallel programming models.

Bibliography

[1] R. Abgrall, T. Sonar, and S. Lanteri. ENO approximations for compressible fluid dynamics. *ZAMM*, 79(1):3–28, 1999.

[2] E.R. Adair and R.C. Petersen. Biological effects of radio-frequency/microwave radiation. *IEEE Trans. Microwave Theory Tech.*, 50(3):953–962, 2003.

[3] M.F. Adams. A parallel Maximal Independent Set algorithm. In *5th Copper Mountain Conference on Iterative Methods*. http://www.mgnet.org/mgnet-cmcim98.html, 1998.

[4] W.K. Anderson and D.L. Bonhaus. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Comput. Fluids*, 23:1–21, 1994.

[5] W.K. Anderson, W.D. Gropp, D.K. Kaushik, D. Keyes, and B. Smith. Achieving high sustained performance in an unstructured mesh CFD application (Bell Prize award paper, Special Category). In *SC'99 High Performance Networking and Computing Conference*, 1999.

[6] F. Angrand and A. Dervieux. Some explicit triangular finite element schemes for the Euler equations. *Int. J. Numer. Meth. Fluids*, 4:749–764, 1984.

[7] P. Arminjon and A. Dervieux. Construction of TVD-like artificial viscosities on two-dimensinal arbitrary FEM grids. *J. Comput. Phys.*, 106:176–198, 1993.

[8] T. Barth. *Numerical methods for gas dynamics systems on unstructured meshes*, volume 5 of *Lecture Notes in Computational Science and Engineering*, pages 195–285. Springer Verlag, 1999.

[9] T. Barth, T.-F. Chan, and W.-P. Tang. A parallel non-overlapping domain decomposition algorithm for compressible fluid flow problems on triangulated domains. In C. Farhat J. Mandel and X.-C. Cai, editors, *10th International Conference on Domain Decomposition Methods in Science and Engineering*, number 218 in Contemporary Mathematics, pages 23–41. AMS, 1998.

[10] P. Bastian, W. Hackbusch, and G. Wittum. Additive and multiplicative multi-grid: a comparison. Technical Report 95-08, Institute for Computer Applications (ICA), 1995.

[11] M. Bjørhus. Semi-discrete subdomain iteration for hyperbolic systems. Technical Report 4, NTNU, 1995.

[12] A. Bouras and V. Fraysse. A relaxing strategy for inexact matrix-vector products for krylov methods. Technical Report TR/PA/00/15, CERFACS, 2000.

[13] A. Bouras and V. Fraysse. A relaxing strategy for inner-outer linear solvers in domain decomposition methods. Technical Report TR/PA/00/17, CERFACS, 2000.

[14] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31:333–390, 1977.

[15] A. Brandt. Guide to multigrid development. In A. Hackbush and U. Trottenberg, editors, *Conference on Multigrid Methods*, number 960 in Lecture notes in Mathematics, pages 220–312, Köln-Porz, Allemagne, 1982.

[16] A. Brandt and B. Diskin. Multigrid solvers on decomposed domains. In Y.A. Kuznetsov A. Quarteroni, J. Périaux and O. Widlund, editors, *6th International Conference on Domain Decomposition Methods in Science and Engineering*, number 157 in Contemporary Mathematics, pages 135–155. AMS, 1994.

[17] E. Briand and S. Lanteri. Development of parallel MG and FMG algorithms for turbulent Navier-Stokes equations. Technical report, IDeMAS technical report, task 3.5.2, 2001.

[18] W.L. Briggs, V.H. Henson, and S.F McCormick. *A multigrid tutorial*. SIAM, 2000.

[19] X.-C. Cai, C. Farhat, and M. Sarkis. Variable degree Schwarz methods for the implicit solution of unsteady compressible Navier-Stokes equations on two-dimensional unstructured meshes. Technical Report 96-48, Institute for Computer Applications in Sciences and Engineering (ICASE), 1996.

[20] X.-C. Cai, C. Farhat, and M. Sarkis. A minimum overlap restricted additive Schwarz preconditioner and appication in 3D flow simulations. In C. Farhat J. Mandel and X.-C. Cai, editors, *10th International Conference on Domain Decomposition Methods in Science and Engineering*, number 218 in Contemporary Mathematics, pages 479–485. AMS, 1998.

[21] X.-C. Cai, D. Keyes, and L. Marcinkowski. Non-linear additive Schwarz preconditioners and applications in computational fluid dynamics. *Int. J. Numer. Meth. Fluids*, 40:1463–1470, 2002.

[22] G. Carré. *Simulation numérique d'écoulements turbulents compressibles stationnaires par méthodes multigrilles*. Thèse Sciences de l'Ingénieur, Université de Nice-Sophia Antipolis, November 1995.

[23] G. Carré. An implicit multigrid method by agglomeration applied to turbulent flows. *Comput. Fluids*, 26:299–320, 1997.

[24] G. Carré and A. Dervieux. On the application of FMG to variational approximation of flow problems. *Comp. Fluid. Dyn. J.*, 12:99–117, 1999.

[25] G. Carré, L. Fournier, and S. Lanteri. Parallel linear multigrid algorithms applied to the acceleration of compressible flows. *Comput. Methods Appl. Mech. Engrg.*, 184:427–448, 2000.

[26] G. Carte, T. Coupez, H. Guillard, and S. Lanteri. Coarsening techniques in multigrid applications on unstructured meshes. In *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS00)*, 2000.

[27] G. Carte and S. Lanteri. Development of parallel MG and FMG algorithms for Euler and Navier-Stokes equations. Technical report, IDeMAS technical report, task 3.5.1, 2000.

[28] T.-F. Chan and B. Smith. Domain decomposition and multigrid techniques for elliptic problems on unstructured meshes. *Electronic Transaction in Numerical Analysis*, 2:171–182, 1994.

[29] T.-F. Chan and R.-S. Tuminaro. Design and implementation of parallel multigrid algorithms. Technical Report 87.21, Research Institute for Advanced Computer Science (RIACS), 1987.

[30] T.-F. Chan and R.-S. Tuminaro. Analysis of a parallel multigrid algorithm. Technical Report 89.41, Research Institute for Advanced Computer Science (RIACS), 1989.

[31] J.P. Cioni, L. Fezoui, and H. Steve. A parallel time-domain Maxwell solver using upwind schemes and triangular meshes. *Impact Comput. Sci. Engrg.*, 215-247:1–21, 1993.

[32] S. Clerc. *Etude de schémas décentrés implicites pour le calcul numérique en mécanique des fluides. Résolution par décomposition de domaine*. Thèse en Analyse Numérique, Université de Paris VI, 1997.

[33] S. Clerc. Non-overlapping Schwarz method for systems of first order equations. In C. Farhat J. Mandel and X.-C. Cai, editors, *10th International Conference on Domain Decomposition Methods in Science and Engineering*, number 218 in Contemporary Mathematics, pages 408–416. AMS, 1998.

[34] T. Coupez and J.-L. Chenot. Mesh topology for mesh generation problem. Application to 3-D remeshing. In J.-L. Chenot and O. Zienkiewicz, editors, *Numerical methods in industrial forming processes*, pages 237–242, 1992.

[35] H. Deconinck and B. Koren, editors. *Euler and Navier-Stokes solvers using multi-dimensional upwind schemes and multigrid acceleration*, volume 57. Vieweg, 1997.

[36] A. Dervieux, D. Guezengar, H. Guillard, and C. Viozat. Analysis of low mach simulations with compressible upwind codes. In J.Périaux K.-D.Papailiou, D. Tsahalis and D. Knorzer, editors, *4th European Computational Fluid Dynamics Conference (ECCOMAS98)*, pages 38–44. John Wiley & Sons, 1998.

[37] A. Dervieux, S. Lanteri, J.-M. Malé, N. Marco, N. Rostaing-Schmidt, and B. Stoufflet. New technologies for advanced three-dimensional optimum shape design in aeronautics. *Int. J. Numer. Meth. Fluids*, 30:179–191, 1999.

[38] J.-A. Désidéri, S. Lanteri, N. Marco, B. Mantel, and J. Périaux. Genetic Algorithms for a two-dimensional shape optimum design problem. *Surv. Math. Ind.*, 9:207–221, 2000.

[39] V. Dolean. *Algorithmes par décomposition de domaine et accélération multigrille pour le calcul d'écoulements compressibles*. Thèse de Mathématiques, Université de Nice-Sophia Antipolis, 2001.

[40] V. Dolean and S. Lanteri. A domain decomposition approach to finite volume solutions of the Euler equations on unstructured triangular meshes. *Int. J. Numer. Meth. Fluids*, 37:625–656, 2001.

[41] V. Dolean, S. Lanteri, and F. Nataf. Construction of interface conditions for solving the compressible Euler equations by non-overlapping domain decomposition methods. *Int. J. Numer. Meth. Fluids*, 40:1485–1492, 2002.

[42] V. Dolean, S. Lanteri, and F. Nataf. Convergence analysis of additive Schwarz for the Euler equations. *Appl. Numer. Math.*, 2003. Accepted for publication.

[43] C.-C. Douglas. A review of numerous parallel multigrid methods. In G. Astfalk, editor, *Applications on advanced architecture computers*, SIAM. Software - Environments - Tools, pages 187–202, Philadelphia, 1996.

[44] B. Engquist and A. Majda. Absorbing boundary conditions for the numerical simulation of waves. *Math. Comp.*, 31:629–651, 1977.

[45] H.K. Engquist, B. Zhao. Absorbing boundary conditions for domain decomposition. *Appl. Numer. Math.*, 27(4):341–365, 1998.

[46] C. Farhat, L. Fezoui, and S. Lanteri. Two-dimensional viscous flow computations on the Connection Machine: unstructured meshes, upwind schemes, and massively parallel computations. *Comput. Methods Appl. Mech. Engrg.*, 102:61–88, 1993.

[47] C. Farhat, M. Lanteri, S.and Lesoinne, and P. Stern. High performance solution of three-dimensional nonlinear aeroelastic problems via parallel partitioned algorithms: methodology and preliminary results. *Adv. Engrg. Softw.*, 28:43–61, 1997.

[48] C. Farhat and S. Lanteri. Simulation of compressible viscous flows on a variety of MPPs: computational algorithms for unstructured dynamic meshes and performance results. *Comput. Methods Appl. Mech. Engrg.*, 119:35–60, 1994.

[49] C. Farhat and M. Lesoinne. Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics. *Internat. J. Numer. Meths. Engrg.*, 36:745–764, 1993.

[50] C. Farhat, H. Simon, and S. Lanteri. TOP/DOMDEC : a software tool for mesh partitioning and parallel processing and applications to CSM and CFD computations. *Comput. Syst. Engrg.*, 6:13–26, 1995.

[51] F. Fezoui. Résolution des équations d'Euler par un schéma de van Leer en éléments finis. Rapport de recherche n RR-358, INRIA, 1985.

[52] L. Fezoui and A. Dervieux. Finite element non-oscillatory schemes for compressible flows. In *8th France-U.S.S.R.-Italy Joint Symposium on Computational Mathematics and Applications*, volume 370 of *IAN*, Pavie, Italie, 1989.

[53] L. Fezoui and S. Lanteri. Parallel upwind FEM for compressible flows. In *Parallel Computational Fluid Dynamics 91 Conference (PCFD'91)*, pages 149–163. Elsevier Science Publishers B.V., North Holland, 1993.

[54] L. Fezoui, S. Lanteri, B. Larrouturou, and C. Olivier. Résolution numérique des équations de Navier-Stokes pour un fluide compressible en maillage triangulaire. Technical Report 1033, INRIA, 1989.

[55] L. Fezoui, S. Lanteri, and M. Loriot. Compressible Navier-Stokes solvers on MPPs. In W.G. Habashi, editor, *International Workshop on Solution Techniques for Large-Scale CFD Problems, CERCA, Montréal, Québec, Canada*, pages 99–121. Computational Methods in Applied Sciences, John Wiley & Sons, 1995.

[56] L. Fezoui and B. Stoufflet. A class of implicit upwind schemes for Euler simulations with unstructured meshes. *J. of Comp. Phys.*, 84:174–206, 1989.

[57] J. Francescatto and A. Dervieux. A semi-coarsening strategy for unstructured multigrid based on agglomeration. *Int. J. Numer. Meth. Fluids*, 26:927–957, 1998.

[58] O. Fredrickson and O. McBryan. *Parallel superconvergent multigrid*, volume 110 of *Lecture Notes in Pure and Applied Mathematics*, pages 195–210. 1988.

[59] D. Gannon and J. Van Rosendale. On the structure of parallelism in a highly concurrent PDE solver. *J. Parallel Distrib. Comput.*, 3:106–135, 1986.

[60] F.-R. Gantmacher. *Théorie des matrices*. Dunod, 1965.

[61] F. Gastaldi and L. Gastaldi. On a domain decomposition for the transport equation : theory and finite element approximation. *IMA J. Numer. Anal.*, 14:111–135, 1993.

[62] F. Gastaldi, L. Gastaldi, and A. Quarteroni. Adaptive domain decomposition methods for advection-dominated equations. *East-West J. Numer. Math.*, 4:165–206, 1996.

[63] P.-L. George, F. Hecht, and E. Saltel. Automatic mesh generator with specified boundary. *Comput. Methods Appl. Mech. and Engrg.*, 92:269–288, 1991.

[64] E. Godlewski and P.-A. Raviart. *Numerical approximation of hyperbolic systems of conservation laws*, volume 118 of *Applied Mathematical Sciences*. Springer Verlag, 1996.

[65] S. Godunov. A difference method for the numerical calculation of discontinuous solutions of the hydrodynamic equations. *Mat. Sbornik.*, 47:271–306, 1959.

[66] D. Gropp, D.K. Kaushik, D. Keyes, and B. Smith. High-performance parallel implicit CFD. *Parallel Comput.*, 27:337–362, 2001.

[67] H. Guillard. Node-nested multigrid with Delaunay coarsening. Technical Report 1898, INRIA, mai 1993.

[68] W. Hackbusch. *Multigrid methods and applications*, volume 4 of *Springer Series in Computationl Mathematics*. Springer Verlag, 1985.

[69] A. Harten. On a class of high resolution total-variation stable finite-difference schemes. *J. of Comp. Phys.*, 49:357–393, 1989.

[70] P.-W Hemker. On the order of prolongations and restrictions in multigrid procedures. *J. Comput. Appl. Math.*, 32:423–429, 1990.

[71] S.-A. Hutchinson, J.S. Shadid, and R.S. Tuminaro. AZTEC user's guide, version 1.1. Technical report, Massively Parallel Computing Research Laboratory, Sandia National Laboratories, 1995.

[72] A. Iollo, A. Dervieux, J.-A. Désidéri, and S. Lanteri. Two stable POD-based approximations to the Navier-Stokes equations. *Computing and Visualization in Science*, 3:61–66, 2000.

[73] A. Iollo, J.-A. Désidéri, and S. Lanteri. Stability properties of POD-Galerkin approximations for the compressible Navier-Stokes equations. *Theoret. Comput. Fluid Dynamics*, 13:377–396, 2000.

[74] E. Issman. *Implicit solution strategies for compressible flow equations on unstructured grids*. PhD thesis, Université Libre de Bruxelles, Belgique, 1997.

[75] A. Jameson. Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence in transonic and hypersonic flows. *AIAA paper*, (93-3359), 1993.

[76] C. Japhet and F. Nataf. The best interface conditions for domain decomposition methods: absorbing boundary conditions. In L. Tourrette, editor, *Artificial Boundary Conditions, with Applications to Computational Fluid Dynamics Problems*, pages 348–373. Nova Science, 2000.

[77] C. Japhet, F. Nataf, and F. Rogier. The Optimized Order 2 method. application to convection-diffusion problems. *Future Generation Computer Systems*, 18:17–30, 2001.

[78] C. Johnson. *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press, 1987.

[79] J. Jones and S. McCormick. Parallel multigrid algorithms. In A. Sameh D. Keyes and V. Venkatakrishnan, editors, *Proceedings of the NASA/LaRC Interdisciplinary Series in Science and Enginering. Parallel Numerical Algorithms*, Hampton, Virginia, USA, 1997. Kluwer.

[80] G. Karypis and V. Kumar. `MeTiS`, unstructured graph partitioning and sparse matrix ordering system, user's guide, version 2.0. Technical report, University of Minnesota, Department of computer science, Minneapolis, 1995.

[81] B. Koobus, M.-H. Lallemand, and Dervieux A. Unstructured volume agglomeration MG: solution of the Poisson equation. *Int. J. Numer. Meth. Fluids*, 18:27–42, 1994.

[82] M.-H. Lallemand, Steve S., and Dervieux A. Unstructured multigridding by volume agglomeration : current status. *Computers & Fluids*, 21:397–433, 1992.

[83] S. Lanteri. *Simulation d'écoulements aérodynamiques instationnaires sur une architecture S.I.M.D. massivement parallèle*. Thèse en Sciences de l'Ingénieur, Université de Nice-Sophia Antipolis, 1991.

[84] S. Lanteri. Parallel solutions of compressible flows using overlapping and non-overlapping mesh partitioning strategies. *Parallel Comput.*, 22:943–968, 1996.

[85] S. Lanteri and M. Loriot. Large-scale solutions of three-dimensional compressible flows using the parallel N3S-MUSCL solver. *Concurrency, Pract. Exp.*, 8:769–798, 1996.

[86] S. Lanteri and N. Marco. A two-level parallelization strategy for Genetic Algorithms applied to optimum shape design. *Parallel Comput.*, 26:377–397, 2000.

[87] M. Loriot. Mesh splitter 3D V3.1 user manual, 1998. Simulog.

[88] J. Périaux M.-O. Bristeau, R. Glowinski and H. Viviand, editors. *GAMM Workshop on the Numerical Simulation of Compressible Navier-Stokes Flows*, volume 18 of *Notes on Numerical Fluid Mechanics*. Wieveg, 1987.

[89] R. Martin and V. Guillard. A second order defect correction scheme for unsteady problems. *Computers & Fluids*, 25:9–27, 1996.

[90] L. Martinelli and A. Jameson. Validation of a multigrid method for the Reynolds averaged Navier-Stokes equations. *AIAA paper*, (88-0414), 1988.

[91] L. R. Matheson. *Multigrid algorithms on massively parallel computers*. Thesis philosophy, Princeton University, 1994.

[92] D. J. Mavriplis. Directional agglomeration multigrid techniques for high-reynolds number viscous flows. Technical Report 98-7, Institute for Computer Applications in Sciences and Engineering (ICASE), 1998.

[93] D. J. Mavriplis. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. Technical Report 98-6, Institute for Computer Applications in Sciences and Engineering (ICASE), 1998.

[94] D. J. Mavriplis and S. Pirzadeh. Large-scale parallel unstructured mesh computations for 3D high-lift analysis. *AIAA paper*, (99-0537), 1999.

[95] D. J. Mavriplis and V. Venkatakrishnan. Agglomeration multigrid for two-dimensional viscous flows. *J. Comput. Phys.*, 24:553–570, 1995.

[96] D. J. Mavriplis and V. Venkatakrishnan. A 3D agglomeration multigrid solver for the Reynolds-averaged Navier-Stokes equations on unstructured meshes. *Int. J. for Num. Meth. in Fluids*, 23:527–544, 1996.

[97] D.J. Mavriplis and A. Jameson. Multigrid solution of the two-dimensional Euler equations on unstructured meshes. *AIAA paper*, (87-0353), 1987.

[98] D.J. Mavriplis and A. Jameson. Accurate multigrid solution of the Euler equations on unstructured and adaptive meshes. *AIAA paper*, (88-3707), 1988.

[99] D.J. Mavriplis, A. Jameson, and L. Martinelli. Multigrid solution of the Navier-Stokes equations on triangular meshes. Technical Report 89-11, Institute for Computer Applications in Sciences and Engineering (ICASE), 1989.

[100] K. Mer. Variational analysis of a mixte finite element/finite volume scheme on general triangulations. Rapport de recherche n RR-2213, INRIA, 1994.

[101] L.M. Mesaros and P.L. Roe. Multidimensional fluctuation splitting schemes based on decomposition methods. *AIAA paper*, (95-1699), 1995.

[102] E. Morano. *Résolution des équations d'Euler par une méthode multigrille stationnaire*. Thèse Sciences de l'Ingénieur, Université de Nice-Sophia Antipolis, December 1992.

[103] E. Morano and A. Dervieux. Steady relaxation methods for unstructured MG Euler and Navier-Stokes solutions. *Comput. Fluids*, 5:137–167, 1995.

[104] E. Morano, A. Dervieux, H. Guillard, M.-P. Leclercq, and B. Stoufflet. Faster relaxations for non-structured MG with Voronoi coarsening. In *First European Computational Fluid Dynamics Conference (CFD'92)*, volume 1, pages 69–74. Elsevier, 1992.

[105] A.-W. Mulder. A new multigrid approach to convection problems. *J. Comput. Phys.*, 83:303–323, 1989.

[106] J.-D. Muller. Coarsening 3D hybrid meshes for multigrid methods. In *Copper Mountain Conference on Multigrid Methods*. http://www.mgnet.org/mgnet-ccmm99.html, 1999.

[107] F. Nataf. On the use of open boundary conditions in block gauss-seidel methods for the convection-diffusion equation. Technical Report RI284, Centre de Mathématiques Appliquées, Ecole Polytechnique, 1993.

[108] F. Nataf. Absorbing boundary conditions in block Gauss-Seidel methods for convection problems. *Math. Models Methods Appl. Sci.*, 6(4):481–502, 1996.

[109] B. N'Konga and H. Guillard. Godunov type method on non-structured meshes for three-dimensional moving boundary problems. *Comp. Meth. in Appl. Mech.*, 113:183–204, 1994.

[110] S. Osher and F. Solomon. Upwind schemes for hyperbolic systems of conservation laws. *Math. Comp.*, 38:339–374, 1982.

[111] H. Paillere. *Multidimensional upwind residual distribution schemes for the Euler and Navier-Stokes equations on unstructured grids*. PhD thesis, Université Libre de Bruxelles, Belgique, 1995.

[112] M. Paraschivoiu, X.-C. Cai, M. Sarkis, D.-P. Young, and D. Keyes. Multi-domain multi-model formulation for compressible flows: conservative interface coupling and parallel implicit solvers for 3D unstructured meshes. *AIAA paper*, (99-0784), 1999.

[113] S. Piperno and L. Fezoui. A centered discontinuous galerkin finite volume scheme for the 3D heterogeneous Maxwell equations on unstructured meshes. Technical Report 4733, INRIA, 2003.

[114] S. Piperno, M. Remaki, and L. Fezoui. A non-diffusive finite volume scheme for the 3D Maxwell equations on unstructured meshes. *SIAM J. Numer. Anal.*, 39(6):2089–2108, 2002.

[115] A. Quarteroni. Domain decomposition methods for systems of conservation laws : spectral collocation approximation. *SIAM J. Sci. Stat. Comput.*, 11:1029–1052, 1990.

[116] A. Quarteroni and L. Stolcis. Homogeneous and heterogeneous domain decomposition methods for compressible flow at high reynolds numbers. Technical Report 33, CRS4, 1996.

[117] A. Quarteroni and A. Valli. *Theory and application of Steklov-Poincaré operators for boundary value problems*, pages 179–203. Applied and Industrial Mathematics. Kluwer Academic Publishers, 1991.

[118] A. Quarteroni and A. Valli. *Domain decomposition methods for compressible flows*, pages 221–245. Error control and adaptivity in scientific computing. Kluwer Academic Publishers, 1999.

[119] A. Quarteroni and A. Valli. *Domain decomposition methods for partial differential equations*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1999.

[120] P.L. Roe. Approximate Riemann solvers, parameter vectors and difference schemes. *J. of Comp. Phys.*, 43:357–371, 1981.

[121] P.L. Roe. Multidimensional upwinding: motivation and concepts. VKI LS 1994-05, Computational Fluid Dynamics, 1984.

[122] A. Rosen, M.A. Stuchly, and Vander Vost A. Applications of RF/microwaves in medicine. *IEEE Trans. Microwave Theory Tech.*, 50(3):963–974, 2003.

[123] P. Rostand and B. Stoufflet. Finite volume Galerkin methods for viscous gas dynamics. Rapport de recherche n RR-863, INRIA, 1988.

[124] Y. Saad and H. Schultz. GMRES: Generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[125] M. Sala. An algebraic 2-level domain decomposition preconditioner with applications to the compressible Euler equations. *Int. J. Numer. Meth. Fluids*, 40:1551–1560, 2002.

[126] K. Sermeus, H. Deconinck, G. Carte, and S. Lanteri. A parallel multigrid accelerated multidimensional upwind solver for 3D high Reynolds number flows on unstructured grids. *AIAA paper*, (01-0137), 2001.

[127] P.L. Sidilkover, D.and Roe. Unification of some advection schemes in two dimensions. Technical Report 95-10, Institute for Computer Applications in Sciences and Engineering (ICASE), 1995.

[128] H. Simon. Partitioning of unstructured problems for parallel processing. *Comput. Sys. Engrg.*, 2:135–148, 1991.

[129] V. Simoncini and D.B. Szyld. Theory of inexact krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Stat. Comput.* To appear.

[130] B. Smith, P. Bjorstad, and W. Gropp. *Domain decomposition and parallel multilevel methods for elliptic partial differential equations*. Cambridge University Press, 1996.

[131] J. Steger and R. F. Warming. Flux vector splitting for the inviscid gas dynamic with applications to finite difference methods. *J. of Comp. Phys.*, 40:263–293, 1981.

[132] B. Stoufflet, J. Priaux, L. Fezoui, and A. Dervieux. Numerical simulation of 3D hypersonic Euler flows around space vehicles using adapted finite elements. *AIAA paper*, (87-06560), 1987.

[133] R. Struijs. *A multi-dimensional upwind discretization method for the Euler equations on unstructured grids*. PhD thesis, University of Delft, The Netherlands, 1994.

[134] R. Struijs, H. Deconinck, and P.L. Roe. Fluctuation splitting schemes for the 2D Euler equations. Technical Report LS 1991-01, Computational Fluid Dynamics, VKI, 1991.

[135] P.K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journ. Num. Anal.*, 21:995–1011, 1989.

[136] M.-D. Tidriri. Hybrid Newton-Krylov domain decomposition methods for compressible flows. In M.S. Espedal P.E. Bjorstad and D. Keyes, editors, *9th International Conference on Domain Decompositon Methods in Science and Engineering*, pages 532–539. John Wiley & Sons, 1998.

[137] E. Toro. *Riemann solvers and numerical methods for fluid dynamics*. Springer, 1997.

[138] U. Trottenberg, C.W. Oosterlee, and A. Schuller. *Multigrid*. Academic Press, 2001.

[139] R.-S. Tuminaro. A highly parallel multigrid-like method for the solution of the Euler equations. *SIAM J. Sci. Stat. Comput.*, 13:88–100, 1992.

[140] E. van der Weide, K. Sermeus, and H. Deconinck. Thor v1.0, basic Euler solver. Technical report, IDeMAS technical report, tasks 2.1/2.2/2.3, 1998.

[141] B. Van Leer. Towards the ultimate conservative difference scheme V : a second-order sequel to Godunov's method. *J. of Comp. Phys.*, 32:361–370, 1979.

[142] B. Van Leer, W.T. Lee, and P.L. Roe. Characteristic time-stepping or local preconditioning of the Euler equations. *AIAA paper*, (91-1552-CP), 1991.

[143] P. Wesseling. *An introduction to multigrid methods*. John Wiley & Sons, 1991.

[144] R. Wienands. Fourier analysis of GMRES(m) preconditioned by multigrid. In *Proceedings of the 9th Copper Mountain Conference on Multigrid Methods*, Copper Mountain, Colorado, USA, 1999.

[145] G. Wittum. Linear iterations as smoothers in multigrid methods: theory with applications to incomplete decompositions. *Impact Comput. Sci. Engrg.*, 1:180–215, 1989.

[146] Y Wu, X.-C. Cai, and D.-E. Keyes. Additive Schwarz methods for hyperbolic equations. In C. Farhat J. Mandel and X.-C. Cai, editors, *10th International Conference on Domain Decomposition Methods in Science and Engineering*, number 218 in Contemporary Mathematics, pages 468–476. AMS, 1998.

[147] D. Xie and R. Scott. The parallel U-cycle multigrid method. Technical report, Department of Mathematics, University of Houston, UH/MD 240, 1997.

[148] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM rev.*, 34:581–613, 1992.

[149] K.S. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Trans. on Antennas and Propagation*, (AP-16):302–307, 1966.

Méthodes numériques performantes en maillages non-structurés et applications en mécanique des fluides compressibles

Résumé

De nos jours, la simulation numérique est couramment utilisée pour étudier des phénomènes physiques de plus en plus complexes. Les puissances de calcul et taille mémoire nécessaires au traitement informatique des systèmes d'équations aux dérivées partielles (EDPs) modélisant ces phénomènes ont conduit à l'utilisation quasi systèmatique de calculateurs parallèles. Afin d'exploiter efficacement les capacités des ces ordinateurs, des travaux de recherche spécifiques doivent être menés qui combinent des aspects allant de la programmation parallèle jusqu'à l'analyse numérique. Il s'agit, d'une part, de développer des programmes informatiques démontrant des accélérations parallèles quasi optimales et, d'autre part, de mettre au point de nouvelles méthodes numériques parfaitement adaptées à ce type d'architecture de calcul. Les travaux présentés ici illustrent une telle démarche dans le contexte de la résolution numérique des systèmes d'EDPs de la mécanique des fluides compressibles. Nous nous intéressons plus particulièrement à des méthodes numériques reposant sur l'utilisation de maillages non-structurés. Nos contributions portent à la fois sur l'évaluation de stratégies de parallélisation pour ce type de méthodes mais aussi sur la mise au point de méthodes de résolution parallèles des systèmes algébriques résultant de la discrétisation des systèmes d'EDPs en question.

Mots-clés: équations d'Euler, équations de Navier-Stokes, élément fini, volume fini, maillage non-structuré, méthode multigrille, méthode de décomposition de domaine, calcul parallèle.

High-performance numerical methods on unstructured meshes with applications to compressible fluid dynamics

Abstract

Nowadays, numerical simulation is routinely used to study physical phenomena of ever increasing complexity. The computational power and memory capacity that are required for the computer treatment of the systems of partial differential equations (PDEs) modeling these phenomena are such that the use of parallel computing platforms has quickly become essential. In order to fully benefit from the capabilities of those computers, specific reserach activities have to be undertaken that address various topics ranging from computer science concerns to more numerical analysis issues. On one hand, the goal is to develop computer programs that demonstrate optimal parallel speedups. On the other hand, it is necessary to design new numerical methods that are well adapted to this type of computer architecture. In this document, we illustrate these two aspects in the context of the numerical solution of the system of PDEs modeling compressible flows. We consider numerical methods that work with unstructured finite element type discretizations of the underlying computational domain. Our contributions are concerned with both the study of parallelization strategies for existing methods and the construction of parallel algorithms for the resolution of the algebraic systems resulting from the discretization of the underlying systems of PDEs.

Keywords: Euler equations, Navier-Stokes equations, finite element, finite volume, unstructured mesh, multigrid method, domain decomposition method, parallel computing.