A Non-Overlapping Domain Decomposition Method for Solving the Navier-Stokes Equations on Unstructured Triangular Meshes

V. Dolean, Stephane Lanteri

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A non-overlapping domain decomposition method for solving the Navier-Stokes equations on unstructured triangular meshes

V. Dolean, S. Lanteri

N° 3977

Juin 2000

THÈME 4

ISSN 0249-6399 ISRN INRIA/RR--3977--FR+ENG

INRIA
SOPHIA ANTIPOLIS

A non-overlapping domain decomposition method for solving the Navier-Stokes equations on unstructured triangular meshes

V. Dolean*, S. Lanteri*

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Sinus

Rapport de recherche n° 3977 — Juin 2000 — 48 pages

Abstract: we report on our recent efforts on the formulation and the evaluation of a non-overlapping domain decomposition method for the parallel solution of two-dimensional compressible viscous flows. This work extends a previous study[11] which was concerned with the design of a domain decomposition solver for the Euler equations discretized on unstructured triangular meshes. As in [11], the method relies on the formulation of an additive Schwarz type algorithm where the interface conditions express the continuity of the normal flux components. The starting point is a flow solver for the Navier-Stokes equations which is based on a combined finite element/finite volume formulation on unstructured triangular meshes for the spatial approximation. Time integration of the resulting semi-discrete equations is performed by using a linearized backward Euler implicit scheme. As a result, each pseudo time step requires the solution of a sparse linear system for the flow variables. In this study, a non-overlapping domain decomposition algorithm is used for advancing the solution at each implicit time step. Algebraically speaking, the Schwarz algorithm is equivalent to a Jacobi iteration applied to a linear system whose matrix has a block structure. A substructuring technique can be applied to this matrix in order to obtain a fully implicit scheme in terms of interface unknowns. In our approach, the interface unknowns are numerical fluxes.

Key-words: Domain decomposition method - Navier-Stokes equations - Finite elements - Finite volumes - Triangular meshes - Multigrid algorithm - Parallel computing

* INRIA, 2004 Route des Lucioles, BP. 93, 06902 Sophia Antipolis Cédex-France

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Une méthode de décomposition de domaine sans recouvrement pour la résolution des équations de Navier-Stokes en maillages triangulaires non-structurés

Résumé : dans ce rapport, on formule et on évalue numériquement une méthode par décomposition de domaine sans recouvrement pour la résolution parallèle d'écoulements bidimensionnels de fluides visqueux. Il s'agit ici de l'extension d'une étude préliminaire[11] qui portait sur la mise au point d'une telle méthode pour la résolution des équations d'Euler (cas d'un fluide parfait) en maillages triangulaires. Comme dans [11], la méthode proposée repose sur la formulation d'un algorithme de type Schwarz additif basé sur des conditions d'interface exrprimant la continuité des flux normaux. Le point de départ de notre étude est constitué d'un solveur des équations de Navier-Stokes qui repose sur une formulation mixte éléments finis/volumes finis en maillages triangulaires pour l'approximation en espace. L'intégration en temps est réalisée au moyen d'un schéma d'Euler implicite linéarisée. Chaque pas de temps nécessite la résolution approchée d'un système linéaire de matrice creuse non-symétrique. Ici, on utilise un algorithme par décomposition de domaine non-recouvrant pour la réalisation d'un pas de temps implicite. D'un point de vue algébrique, l'algorithme de Schwarz peut être interprété comme une méthode de relaxation de Jacobi appliquée à la résolution d'un système linéaire dont la matrice a une structure par bloc particulière. Une technique de sous-structuration peut être appliquée à cette matrice afin d'obtenir un système interface. Dans notre cas, les variables d'interface sont des flux numériques. Il en résulte un algorithme par décomposition de domaine du type complément de Schur.

Mots-clés : Méthode de décomposition de domaine - Equations de Navier-Stokes - Eléments finis - Volumes finis - Maillages triangulaires - Algorithme multigrille - Calcul parallèle

Contents

1 Introduction

When solving a problem modelled by a partial differential equation, one is generally confronted to a discretization step followed by a linear system solve (or a series of linear system solves). The size or the ill-conditioning of these systems makes a global or a direct solution approach quite inappropriate. On the other hand, with the advent of parallel computers, domain decomposition algorithms have enjoyed an increasing popularity among the scientific community because they define a good framework to derive efficient solvers for the resulting systems using the mathematical properties of the initial PDE[32]. Indeed, since the early 1980's, efficient and scalable domain decomposition algorithms have been developed for the solution of computational structural mechanics problems (i.e. for the solution of elliptic PDEs); however their application to computational fluid dynamics problems (i.e. to the solution of hyperbolic or mixed hyperbolic/parabolic PDEs) has been less remarkable.

Domain decomposition methods are generally classified according to two criteria :

- *overlapping* versus *non-overlapping* methods according to the spatial decomposition of the global domain;
- *multiplicative* versus *additive* methods according to the interdependence of the local solutions at each iteration.

The non-overlapping domain decomposition methods can be of the Schwarz or of the substructuring (Schur complement or interface system) type, the latest being related to block Gaussian elimination techniques (each block corresponding to a different subdomain). At the continuous level, in the context of a substructuring method, one has to deal with an operator acting on interface variables whose discretization is the Schur complement of the global operator[27],[25].

Domain decomposition methods were first developed for elliptic second-order problems, taking advantage of the strong regularity of their solutions as well as of the symmetry (or the dominance of the symmetric part) of the operators involved[38],[7],[31]. The situation is less clear for hyperbolic or mixed hyperbolic/parabolic models of compressible fluid mechanics. One has to deal with first-order (e.g. the Euler equations) or second-order (e.g. the Navier-Stokes equations) systems of PDEs characterized by non-symmetric operators, with possible singular solutions. When the symmetric part is dominant one can still apply the algorithms built for the symmetric systems with a few modifications. If not, for example when convection is dominant in the convection-diffusion case, different approaches exist using Dirichlet and/or Neumann interface

conditions as in [10], or using a Robin transmission condition and an *iteration by subdomain* algorithm as in [8], [16] and [17].

The objective of the present work is to solve the Navier-Stokes equations for compressible flows by a non-overlapping domain decomposition method, and more precisely by a substructuring method. This work extends a previous study[11] which was concerned with the design of a domain decomposition solver for the Euler equations discretized on unstructured triangular meshes. As in [11], the method relies on the formulation of an additive Schwarz type algorithm where the interface conditions express the continuity of the normal flux components. However, in the present case, the corresponding flux vector combines convective and diffusive terms. The proposed extension consists in applying a separate treatment to these two terms; in particular, for the convective flux vector, Dirichlet conditions are imposed on characteristic variables corresponding to incoming waves. Such a framework has been adopted by several authors (see for instance Quarteroni and Stolcis[24]).

The concrete implementation of the method is done in the context of a flow solver which is based on a combined finite element/finite volume formulation on unstructured triangular meshes for the spatial discretization. Time integration of the resulting semi-discrete equations is obtained using a linearized backward Euler implicit scheme[15]. As a result, each pseudo time step requires the solution of a sparse linear system for the flow variables. In this work, the non-overlapping domain decomposition algorithm is used for advancing the solution at each implicit time step. Algebraically speaking, the Schwarz algorithm is equivalent to a Jacobi iteration applied to a linear system whose matrix has a block structure. A substructuring technique can be applied to this matrix in order to obtain a fully implicit scheme in terms of interface unknowns. In our approach, the interface unknowns are numerical fluxes; more precisely, the vector of interface unknowns is composed of discrete convective fluxes computed on interface edges using the approximate Riemann solver of Roe[28] on one hand, and, on the other hand, of discrete diffusive fluxes computed on interface triangles.

Related works for the solution of the Euler or Navier-Stokes equations can be classified as follows :

- development of algebraic preconditioners based on additive Schwarz formulations for Krylov iterative methods (e.g. Newton-Krylov-Schwarz methods) [1]-[4]-[5]-[6]-[34]-[37].
- development of domain decomposition solvers based on appropriate formulations in the continuous case [24]-[26]-[23]. The present stuty follows this approach.

The paper is organized as follows. In section 2, we recall the mathematical model used for the simulation of compressible viscous flows and we introduce the Schwarz type algorithm in the continuous case. Section 3 describes the characteristics of the starting point mixed finite element/finite volume solver for the Navier-Stokes equations. The proposed domain decomposition approach is then adapted to the discrete case in section 4.

For steady flow calculations, the linear system resulting from the implicit scheme adopted in the original solver is generally solved to a low accuracy (in practice, approximate solutions are obtained using several sweeps of a relaxation method such as the Jacobi or the Gauss-Seidel method). In [11] we have considered using a somewhat similar approach for the domain decomposition solver in the sense that the local solves involved in the formation of the matrix-vector product with the interface operator are performed approximately. These matrix-vector products are induced by the use of a full GMRES method for the solution of the interface system. A particularity of our approach relies in the adoption of a multigrid strategy by volume agglomeration[20] for the local solves. As a result, the proposed domain decomposition solver can also be viewed as a mean of coordination of concurrent multigrid acceleration applied at the subdomain level. This approach has also been adopted in the present study. This is described in section 5.

In section 6, the resulting domain decomposed flow solver is evaluated through numerical experiments that are performed on a cluster of `Pentium Pro` computers interconnected via a 100 Mbit/s `FastEthernet` switch. Finally, conclusions and future works are presented in section 7.

2 Mathematical model and domain decomposition

2.1 The Navier-Stokes equations

Let $\Omega \subset \mathbb{R}^2$ be the computational domain and Γ its boundary. Γ is assumed to be constructed as the union of a solid wall Γ_w and a far-field boundary Γ_∞ : $\Gamma = \Gamma_w \cup \Gamma_\infty$. Let $\vec{n}$ denote the unitary normal at any point of Γ. The conservative form of the Navier Stokes equations is given by :

$$\frac{\partial W}{\partial t} + \frac{\partial F(W)}{\partial x} + \frac{\partial G(W)}{\partial y} = \frac{1}{\mathrm{Re}} \left(\frac{\partial R(W)}{\partial x} + \frac{\partial S(W)}{\partial y} \right) \tag{1}$$

where :

- $\vec{x}$ and t respectively denote the spatial and temporal variables and $W = W(\vec{x}, t) = (\rho\ ,\ \rho u\ ,\ \rho v\ ,\ E)^T$ is the vector of conservative variables;
- $\vec{\mathcal{F}}(W) = (F(W)\ ,\ G(W))^T$ is the convective flux whose components are given by :

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

- $\vec{\mathcal{R}}(W) = (R(W)\ ,\ S(W))^T$ is the viscous flux whose components are given by :

$$R(W) = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\,\tau_{xx} + v\,\tau_{xy} + \dfrac{\gamma\,\mu}{\mathrm{Pr}}\dfrac{\partial e}{\partial x} \end{pmatrix} \quad , \quad S(W) = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\,\tau_{xy} + v\,\tau_{yy} + \dfrac{\gamma\,\mu}{\mathrm{Pr}}\dfrac{\partial e}{\partial y} \end{pmatrix} \tag{3}$$

In the above expressions, ρ is the density, $\vec{U} = (u, v)^T$ is the velocity vector, E and e respectively denote the total energy per unit of volume and the specific internal energy; p is the pressure; p and e are deduced from the other variables using the state equations for a perfect gaz :

$$\begin{cases} p &= (\gamma - 1)(E - \dfrac{1}{2}\rho \parallel \vec{U} \parallel^2) \\ e &= \dfrac{E}{\rho} - \dfrac{1}{2}(u^2 + v^2) = C_v T \end{cases}$$

where T is the temperature and C_v the specific heat coefficient. In the expressions for the diffusive fluxes,τ_{xx}, τ_{xy} and τ_{yy} stand for the components of the Cauchy stress tensor :

$$\begin{cases} \tau_{xx} &= \dfrac{2}{3}\mu\left(2\dfrac{\partial u}{\partial x} - \dfrac{\partial v}{\partial y}\right) \\ \tau_{yy} &= \dfrac{2}{3}\mu\left(2\dfrac{\partial v}{\partial y} - \dfrac{\partial u}{\partial x}\right) \\ \tau_{xy} &= \mu\left(\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial x}\right) \end{cases}$$

where γ is the ratio of specific heat coefficients ($\gamma = 1.4$ for the air), μ and k respectively denote normalized viscosity and conductivity coefficients. Two dimensionless numbers, the Reynolds number and the Prandtl number, appear in the above expressions :

$$\left\{ \begin{array}{rcl} \mathrm{Re} & = & \dfrac{\rho_0 U_0 L_0}{\mu_0} \\ \mathrm{Pr} & = & \dfrac{\mu_0 C_p}{k_0} \end{array} \right.$$

where the subscript 0 is used to denote characteristic values for the flow under consideration and C_p is the specific heat coefficient. Boundary conditions for system (1) are discussed in subsection 3.2.

2.2 Domain decomposition algorithm formulation

The formulation of a domain decomposition algorithm in the continuous case is directly taken from Quarteroni and Stolcis[24]. 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; then, the solution of Eq. (1) can be refomulated as :

$$\left\{ \begin{array}{lll} \dfrac{\partial W}{\partial t} + \vec{\nabla}.\vec{\mathcal{F}}(W) = \dfrac{1}{\mathrm{Re}}\vec{\nabla}.\vec{\mathcal{R}}(W) & \text{for} & \vec{x} \in \Omega_i \\ \left[\left(\vec{\mathcal{F}}(W) - \dfrac{1}{\mathrm{Re}}\vec{\mathcal{R}}(W) \right) .\vec{n}_{ij} \right]_{\Gamma_{ij}} = 0 & \text{for} & \vec{x} \in \Gamma_{ij} \end{array} \right. \tag{4}$$

where $\vec{\mathcal{F}}(W) = (F(W)\ ,\ G(W))^T$ and $\vec{\mathcal{R}}(W) = (R(W)\ ,\ S(W))^T$; $\vec{n}_{ij}$ denotes the normal vector at every point of Γ_{ij}; moreover, $[a]_{\Gamma_{ij}}$ stands for the jump of the quantity a at 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) (see section 3 for a description of the particular variational formulation adopted in this study) and so are the local solutions $W_{|\Omega_i}$.

3 Characteristics of the flow solver

3.1 Spatial approximation method

The flow domain Ω is discretized by a triangulation $\mathcal{T}_h$ 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 $N(i)$. We associate to each vertex s_i a control surface (or cell) denoted by

C_i which is constructed as the union of local contributions from the set of triangles sharing s_i. The contribution of a given triangle is obtained by joining its barycenter G to the midpoints I of the edges incident to s_i (see Fig. 1). The boundary of C_i is denoted by ∂C_i and the unitary normal vector exterior to ∂C_i by $\vec{\nu}_i = (\nu_{ix}, \nu_{iy})$. The union of all these cells constitutes a discretization of Ω often qualified as dual to $\mathcal{T}_h$:

$$\Omega_h = \bigcup_{i=1}^{N_V} C_i \quad , \quad N_V : \text{number of vertices of } \mathcal{T}_h$$

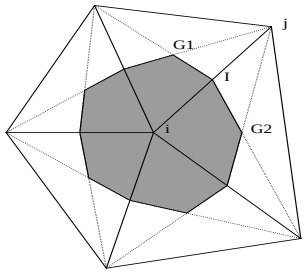

Figure 1: A control surface on a triangular mesh

The spatial discretization method adopted here combines the following ingredients :

- a finite volume formulation together with upwind schemes for the discretization of the convective fluxes. In this formulation, second order accuracy is obtained by using the MUSCL (Monotonic Upstream Schemes for Conservation Laws) technique introduced by van Leer[35] and extended to unstructured triangular meshes by Fezoui and Stoufflet[15];
- a classical Galerkin finite element formulation for the discretization of the diffusive fluxes.

A variational formulation of Eq. (1) can be written as :

$$\iint_{S_i} \left(W_t + \frac{\partial F(W)}{\partial x} + \frac{\partial G(W)}{\partial y} \right) \Psi_i d\vec{x} = \frac{1}{\mathrm{Re}} \iint_{S_i} \left(\frac{\partial R(W)}{\partial x} + \frac{\partial S(W)}{\partial y} \right) \Psi_i d\vec{x} \qquad (5)$$

where S_i is the support of the test function Ψ_i. The test function is choosen differently according to the considered flux :

- convective flux : the test functions is the caracteristic function of the control surface C_i :

$$\Psi_i = \chi\left(C_i\right) \ , \quad S_i = C_i$$

- diffusive flux : the test function is the P_1 basis function (linear and continuous) associated to the vertex s_i :

$$\Psi_i = \phi_i \ \text{ and } \ S_i = \bigcup_{\tau \ , \ s_i \in \tau} \tau \equiv K(i) \ \text{ with } \ \phi_i\left(s_j\right) = \delta_{ij}$$

The system resulting from the application of the Green-Ostrogadsky's formula can be written as :

- for the convective terms

$$\begin{aligned} \iint_{C_i} \left(\frac{\partial F(W)}{\partial x} + \frac{\partial G(W)}{\partial y} \right) \chi_i dx dy &= \int_{\partial C_i} \left(F\left(W\right) \eta_x + G\left(W\right) \eta_y \right) \chi_i dl \\ &- \iint_{C_i} \left(F\left(W\right) \frac{\partial \chi_i}{\partial x} + G\left(W\right) \frac{\partial \chi_i}{\partial y} \right) d\vec{x} \end{aligned}$$

In the above expression, the second term of the right hand side is equal to zero since χ_i is constant over the cell C_i;

- for the viscous terms

$$\begin{aligned} \iint_{K(i)} \frac{1}{\mathrm{Re}} \left(\frac{\partial R(W)}{\partial x} + \frac{\partial S(W)}{\partial y} \right) \phi_i d\vec{x} &= \frac{1}{\mathrm{Re}} \int_{K(i) \cap \Gamma} \left(R\left(W\right) n_x + S\left(W\right) n_y \right) \phi_i dl \\ &- \frac{1}{\mathrm{Re}} \iint_{K(i)} \left(R\left(W\right) \frac{\partial \phi_i}{\partial x} + S\left(W\right) \frac{\partial \phi_i}{\partial y} \right) d\vec{x} \end{aligned}$$

We obtain the discrete equation associated to vertex s_i by calculating :

- the convective flux on the boundary ∂C_i,
- the viscous flux on the support of the basis function ϕ_i, and by considering that the viscous flux on the boundary Γ_∞ is null. This hypothesis, as it happens for most of the external flows, is verified if the boundary of Γ_∞ is far enough of the obstacle (i.e. of the Γ_w boundary). On Γ_w, a strong formulation is applied (see subsection 3.2.1) and the corresponding boundary term is not appearing in the variational formulation.

Then, Eq. (5) becomes :

$$\begin{aligned} \iint_{S_i} W_t \Psi_i d\vec{x} \;\; &+ \;\; \int_{\partial C_i} \vec{\mathcal{F}}(W).\vec{\eta} dl \\ &= \;\; -\frac{1}{\mathrm{Re}} \sum_{\tau \in K(i)} \iint_\tau \left(R(W) \frac{\partial \phi_i}{\partial x} + S(W) \frac{\partial \phi_i}{\partial y} \right) d\vec{x} \end{aligned} \tag{6}$$

where $\vec{\mathcal{F}}(W) = (F(W)\ ,\ G(W))^T$.

3.1.1 Calculation of the convective fluxes

The second term of the left-hand side of Eq. (6) can be decomposed as :

$$\begin{aligned} \int_{\partial C_i} \vec{\mathcal{F}}(W).\vec{\eta} dl \;\; &= \;\; \sum_{j \in N(i)} \int_{\partial C_{ij}} \vec{\mathcal{F}}(W).\vec{\nu}_i dl & < 1 > \\ &+ \int_{\partial C_i \cap \Gamma_w} \vec{\mathcal{F}}(W).\vec{n}_i dl + \int_{\partial C_i \cap \Gamma_\infty} \vec{\mathcal{F}}(W).\vec{n}_i dl & < 2 > \end{aligned} \tag{7}$$

where $\partial C_{ij} = \partial C_i \cap \partial C_j$. Term $< 1 >$ of the right-hand side of Eq. (7) is an assembly process on elementary internal fluxes which are computed as :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) \approx \int_{\partial C_{ij}} \vec{\mathcal{F}}(W).\vec{\nu}_i dl \quad , \quad \vec{\nu}_{ij} = \int_{\partial C_{ij}} \vec{\nu}_i dl \tag{8}$$

where $\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij})$ is the so-called numerical flux function. A conservative scheme is obtained if for any edge $[s_i, s_j]$ the following condition is verified :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) = -\Phi_{\mathcal{F}}(W_j, W_i, \vec{\nu}_{ji})$$

Upwinding is introduced in the calculation of (8) by using the approximate Riemann solver of Roe[28] which gives :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) = \frac{\vec{\mathcal{F}}(W_i) + \vec{\mathcal{F}}(W_j)}{2}.\vec{\nu}_{ij} - \mid \mathcal{A}_R(W_i, W_j, \vec{\nu}_{ij}) \mid \frac{(W_j - W_i)}{2} \tag{9}$$

where :

$$\mathcal{A}_R(W_i, W_j, \vec{\nu}_{ij}) = \left(\frac{\partial \vec{\mathcal{F}}}{\partial W}(W_i, W_j, \vec{\nu}_{ij}).\vec{\nu} \right)_R$$

is the so-called matrix of Roe that verifies the following property :

$$\mathcal{A}_R(W_i, W_j, \vec{\nu}_{ij})(W_j - W_i) = \mathcal{F}(W_j, \vec{\nu}_{ij}) - \mathcal{F}(W_i, \vec{\nu}_{ij})$$

with :

$$\mathcal{F}(W, \vec{\nu}_{ij}) = \vec{\mathcal{F}}(W).\vec{\nu}_{ij}$$

The numerical flux (9) can thus be reformulated as :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) = \mathcal{F}(W_j, \vec{\nu}_{ij}) - \mathcal{A}_R^+(W_i, W_j, \vec{\nu}_{ij})(W_j - W_i)$$

or as :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) = \mathcal{F}(W_i, \vec{\nu}_{ij}) + \mathcal{A}_R^-(W_i, W_j, \vec{\nu}_{ij})(W_j - W_i)$$

with :

$$\mathcal{A}_R(W_i, W_j, \vec{\nu}_{ij}) = A_{\vec{\nu}_{ij}}(\tilde{W})$$

where $\tilde{W}$ is given by :

$$\left\{ \begin{array}{lcl} \tilde{W} & = & (\tilde{\rho}, \tilde{\rho}\tilde{u}, \tilde{\rho}\tilde{v}, \tilde{E})^T \\ \tilde{\rho} & = & (\sqrt{\rho_1}\rho_1 + \sqrt{\rho_2}\rho_2)/(\sqrt{\rho_1} + \sqrt{\rho_2}) \\ \tilde{u} & = & (\sqrt{\rho_1}u_1 + \sqrt{\rho_2}u_2)/(\sqrt{\rho_1} + \sqrt{\rho_2}) \\ \tilde{v} & = & (\sqrt{\rho_1}v_1 + \sqrt{\rho_2}v_2)/(\sqrt{\rho_1} + \sqrt{\rho_2}) \\ \tilde{H} & = & (\sqrt{\rho_1}H_1 + \sqrt{\rho_2}H_2)/(\sqrt{\rho_1} + \sqrt{\rho_2}) \end{array} \right. \tag{10}$$

where $H = \dfrac{\gamma p}{(\gamma - 1)\rho} + \dfrac{u^2 + v^2}{2}$ is the total enthalpy per unit of volume.

The diagonalization of the matrix $\mathcal{A}$ is given by :

$$\left\{ \begin{array}{lcl} \mathcal{A}(\tilde{W}) & = & \mathcal{T}(\tilde{W})\Lambda(\tilde{W})\mathcal{T}^{-1}(\tilde{W}) \\ \Lambda & = & \mathrm{diag}\left(\vec{U}.\vec{\nu} - c\ ,\ \vec{U}.\vec{\nu}\ ,\ \vec{U}.\vec{\nu}\ ,\ \vec{U}.\vec{\nu} + c\right) \end{array} \right.$$

where $\mathcal{T}(\tilde{W}^n)$ is the matrix whose columns are the associated left eigenvectors. We can define the negative and positive parts of $\mathcal{A}$ in the following way :

$$\left\{ \begin{array}{lcl} \mathcal{A}^{\pm}(\tilde{W}) & = & \mathcal{T}(\tilde{W})\Lambda^{\pm}(\tilde{W})\mathcal{T}^{-1}(\tilde{W}) \\ \Lambda & = & diag\left(\lambda_k^{\pm}\right),\ k = 1,4 \\ \lambda_k^{+} & = & \max(\lambda_k, 0) \\ \lambda_k^{-} & = & \min(\lambda_k, 0) \end{array} \right.$$

where $c = \sqrt{\gamma \dfrac{p}{\rho}}$ denotes the speed of sound.

The numerical calculation of the convective flux using Eq. (9) is first order accurate in space. The MUSCL technique[35] for the extension to second order accuracy relies on a linear interpolation of the state vectors W_{ij} and W_{ji} at the interface between the cells C_i and C_j :

$$\tilde{W}_{ij} = \tilde{W}_i + \frac{1}{2}(\vec{\nabla}\tilde{W})_i.s_i\vec{s}_j \quad , \quad \tilde{W}_{ji} = \tilde{W}_j - \frac{1}{2}(\vec{\nabla}\tilde{W})_j.s_i\vec{s}_j \tag{11}$$

where $\tilde{W} = \left(\rho\ ,\ \vec{U}\ ,\ p\right)^T$ — in other words, the interpolation is done using the physical variables instead of the conservative variables. Then, the interpolated states (11) are used as arguments to the numerical flux function (9). The nodal gradients $(\vec{\nabla}\tilde{W})_i$ are obtained from a weighted average of the P1 Galerkin (centered) gradients computed on each triangle of the finite element support of s_i :

$$\left(\vec{\nabla}\tilde{W}\right)_i = \frac{\displaystyle\iint_{C_i} \vec{\nabla}\tilde{W}|_\tau d\vec{x}}{\displaystyle\iint_{C_i} d\vec{x}} = \frac{1}{\mathrm{area}(C_i)} \sum_{\tau \epsilon C_i} \frac{\mathrm{area}(\tau)}{3} \sum_{k=1, k\epsilon\tau}^{3} \tilde{W}_k \vec{\nabla} N_k^\tau \tag{12}$$

where $N_k^\tau(x, y, z)$ is the P1 basis function defined at the vertex s_k and associated with the triangle τ. The construction given by Eq. (9)-(11)-(12) 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 use of a slope limitation procedure[13].

3.1.2 Calculation of the viscous fluxes

The calculation of the viscous fluxes calls for a P1 finite element approximation of the corresponding terms in Eq. (6); more precisely, by assuming that the the physical variables are linear on a triangle τ, one obtain that :

$$\begin{aligned} \iint_{\tau} \left(R\left(W\right) \frac{\partial \phi_i}{\partial x} + S\left(W\right) \frac{\partial \phi_i}{\partial y} \right) d\vec{x} &= \text{area}(\tau) \left(R\left(\tau\right) \frac{\partial \phi_i}{\partial x}|_{\tau} + S\left(\tau\right) \frac{\partial \phi_i}{\partial y}|_{\tau} \right) \\ &= \Upsilon_{\tau,i}(\tau) \end{aligned} \tag{13}$$

where $R\left(\tau\right)$ and $S\left(\tau\right)$ are constant vectors computed on the triangle τ 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).

3.2 Boundary conditions

3.2.1 Wall boundary

Dirichlet type conditions are applied on Γ_w. On one hand, a classical no-slip condition is taken into account for the velocity vector :

$$\vec{U} = 0$$

and, on the other hand, an isotherm condition is considered for the temperature :

$$T = T_w$$

where T_w is computed as :

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

where M_{∞} is the far-field Mach number. The above conditions are applied in a strong way : for a given node s_i of Γ_w, $\vec{U}(s_i)$ is set to 0 and $T(s_i)$ to T_w; the density ρ_i is kept unchanged since the corresponding equation is hyperbolic (in other words, a null mass flux is imposed on $\partial C_i \cap \Gamma_w$). Finally, the total energy per unit of volume and the pressure are updated as :

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

3.2.2 Far-field boundary

On Γ_∞, we assume that the flow is uniform (this assumption is valid for external flows) :

$$\rho_\infty = 1 \quad , \quad \vec{U}_\infty = (u_\infty \ , \ v_\infty)^T \quad \text{with} \quad \| \vec{U}_\infty \| = 1 \quad , \quad p_\infty = \frac{1}{\gamma M_\infty^2} \tag{14}$$

Here, an *upwind-downwind* flux decomposition is used to compute the corresponding boundary integral in Eq. (7). More precisely, this boundary term is evaluated using a non-reflexive version of the Steger and Warming flux decomposition[33] :

$$\int\limits_{\partial C_i \cap \Gamma_\infty} \vec{\mathcal{F}}(W).\vec{n}_i dl = \Phi^{SW}\,(W_i^n, W_\infty, \vec{n}_{i\infty}) = \mathcal{A}^+(W_i, \vec{n}_{i\infty})W_i + \mathcal{A}^-(W_i, \vec{n}_{i\infty})W_\infty \tag{15}$$

3.3 Time integration

Assuming $W(\vec{x}, t)$ is constant on each cell C_i (in other words a mass lumping technique is applied to the temporal term in Eq. (5)) we obtain the following set of semi-discrete equations :

$$\text{area}(C_i)\frac{dW_i^n}{dt} + \Psi(W_i^n) = 0 \quad , \quad i = 1, \cdots, N_V \tag{16}$$

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

$$\Psi(W_i^n) = \sum_{j \in N(i)} \Phi_{\mathcal{F}}(W_{ij}, W_{ji}, \vec{\nu}_{ij}) + \int\limits_{\partial C_i \cap \Gamma_\infty} \vec{\mathcal{F}}(W).\vec{n}_i dl + \frac{1}{\text{Re}} \sum_{\tau \in K(i)} \Upsilon_{\tau,i} \tag{17}$$

Explicit time integration procedures for the time integration of (16)-(17) are subject to a stability condition expressed in terms of a CFL number. On the other hand, an efficient time advancing strategy can be obtained by means of an implicit linearized formulation such as the one described in Fezoui and Stoufflet[15] and briefly outlined here. First, an implicit variant of Eq. (16) writes as :

$$\frac{\text{area}(C_i)}{\Delta t^n}\delta W_i^{n+1} + \Psi(W_i^{n+1}) = 0 \quad , \quad i = 1, \cdots, N_V \tag{18}$$

where $\delta W_i^{n+1} = W_i^{n+1} - W_i^n$. Then, applying a first order linearization to the nodal flux $\Psi(W_i^{n+1})$ yields the Newton-like formulation :

$$\left(\frac{\text{area}(C_i)}{\Delta t^n} + \frac{\partial \Psi(W^n)}{\partial W}\right) \delta W^{n+1} = -\Psi(W_i^n) \tag{19}$$

In practice we replace the exact Jacobian of the second order flux $\frac{\partial \Psi(W^n)}{\partial W}$ by an approximate Jacobian matrix $J(W^n)$ (see the next subsections for more details) and we obtain the following linear system :

$$P(W^n)\delta W^{n+1} = \left(\frac{\text{area}(C_i)}{\Delta t^n} + J(W^n)\right) \delta W^{n+1} = -\Psi(W_i^n) \tag{20}$$

The resulting Euler implicit time integration scheme is in fact a modified Newton method. As a consequence, one cannot ensure that this formulation will yield a quadratically converging method for time steps tending to infinity. The matrix $P(W^n)$ is sparse and has the suitable properties (diagonal dominance in the scalar case) allowing the use of a relaxation procedure (Jacobi or Gauss-Seidel) in order to solve the linear system of Eq. (20). Moreover, an efficient way to get second order accurate steady solutions while keeping the interesting properties of the first order Jacobian matrix is to use the second order elementary convective fluxes based on Eq. (9)-(11)-(12) in the right-hand side of Eq. (20). The above implicit time integration technique is well suited to steady flow calculations; for unsteady flow computations, this first order time accurate scheme is generally unacceptably dissipative.

3.3.1 Linearization of the convective terms

The contribution of the convective terms to the Jacobian matrix is based on an approximate linearization of the first order convective flux (9). For an edge $[s_i, s_j]$ the implicit version of the associated numerical flux is formally written as :

$$\Phi_{ij}^{n+1} = \Phi_{\mathcal{F}}(W_i^n, W_j^n, W_i^{n+1}, W_j^{n+1}, \vec{\nu}_{ij})$$

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

$$\Phi_{\mathcal{F}}(U, V, W, Z, \vec{\nu}_{ij}) = \Phi_{\mathcal{F}}(U, V, \vec{\nu}_{ij}) + \left(\frac{\partial \Phi}{\partial U}\right)(W - U) + \left(\frac{\partial \Phi}{\partial V}\right)(Z - V) \tag{21}$$

where :

$$\Phi_{\mathcal{F}}(U, V, \vec{\nu}_{ij}) = \Phi_{\mathcal{F}}(W_i^n, W_j^n, \vec{\nu}_{ij})$$

is the explicit flux. Expression (21) can be simplified in the case where the numerical flux function takes the form :

$$\Phi_{\mathcal{F}}(U,V,\vec{\nu}_{ij}) = H_1(U,V,\vec{\nu}_{ij})U + H_2(U,V,\vec{\nu}_{ij})V$$

For instance, for the numerical flux function associated to the approximate Riemann solver of Roe[28] we have :

$$\begin{aligned}\Phi_{\mathcal{F}}(W_i,W_j,\vec{\nu}_{ij}) &= \mathcal{F}(W_i,\vec{\nu}_{ij}) + \mathcal{A}_R^-(W_i,W_j,\vec{\nu}_{ij})(W_j - W_i)\\ &= \mathcal{A}(W_i,\vec{\nu}_{ij})W_i + \mathcal{A}_R^-(W_i,W_j,\vec{\nu}_{ij})(W_j - W_i)\end{aligned}$$

therefore :

$$\left\{\begin{array}{lcl} H_1(U,V,\vec{\nu}_{ij}) &=& \mathcal{A}(U,\vec{\nu}_{ij}) - \mathcal{A}_R^-(U,V,\vec{\nu}_{ij})\\ H_2(U,V,\vec{\nu}_{ij}) &=& \mathcal{A}_R^-(U,V,\vec{\nu}_{ij})\end{array}\right.$$

In other words, we obtain an approximate linearization (an thus an approximate Jacobian matrix) if we assume :

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

Note that for the Steger and Warming[33] numerical flux function (see Eq. (15)) we have :

$$\left\{\begin{array}{lcl} H_1(U,V,\vec{\nu}_{ij}) &=& \mathcal{A}^+(U,\vec{\nu}_{ij})\\ H_2(U,V,\vec{\nu}_{ij}) &=& \mathcal{A}^-(V,\vec{\nu}_{ij})\end{array}\right.$$

3.3.2 Linearization of the diffusive terms

As previously, the implicit version of the diffusive flux (13), expressing the contribution of the triangle τ to the global nodal flux associated to the vertex s_i, is formally 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} and s_{k3} are the vertices of the triangle τ (one of them being s_i). Proceeding as for the convective terms we obtain :

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

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

$$\begin{aligned}\Upsilon_{\tau,i}\left(W_{k1}^n, W_{k2}^n, W_{k3}^n\right) &= \text{area}\left(\tau\right)\left(R\left(W_{k1}^n, W_{k2}^n, W_{k3}^n\right)\frac{\partial \phi_i}{\partial x}\mid_\tau + \right. \\ &\left. S\left(W_{k1}^n, W_{k2}^n, W_{k3}^n\right)\frac{\partial \phi_i}{\partial y}\mid_\tau\right)\end{aligned} \tag{22}$$

Therefore the linearization of the viscous flux can also be written as :

$$\Upsilon_{\tau,i}^{n+1} = \Upsilon_{\tau,i}^n + \left(\frac{\partial \Upsilon_{\tau,i}}{\partial W_i}\right)\delta W_i^{n+1} + \sum_{k\in\tau, k\neq i}\left(\frac{\partial \Upsilon_{\tau,i}}{\partial W_k}\right)\delta W_k^{n+1} \tag{23}$$

The derivative terms in Eq. (23) can be computed exactly from the expression of the discretized diffusive flux components of Eq. (22) (see [14] for more details).

3.3.3 Numerical resolution

Taking into account the various contributions of the previous subsections, the implicit integration of the Navier-Stokes equations proceeds in the following way :

- « Physical » or explicit phase : evaluation of the right hand side $\delta\hat{W}_i$ of the linear system to be solved :

$$\delta\hat{W}_i = -\left(\sum_{j\in N(i)}\Phi_{ij}^n + \int_{\partial C_i\cap\partial\Gamma_\infty}\Phi^{SW}\left(W_i^n, W_\infty, \vec{\eta}\right) + \frac{1}{\text{Re}}\sum_{\tau\in K(i)}\Upsilon_{\tau,i}^n\right) \tag{24}$$

- « Mathematical » or implicit phase : the assembling of the different contributions leads to the linear system $M\left(W^n\right)\delta W^{n+1} = \delta\hat{W}$ which is approximately solved by a relaxation method. The matrix $M\left(W^n\right)$ is a non-symmetric sparse matrix. Each term of this matrix is a 4×4 dense block. The i-th line of the linear system writes as :

$$M_{ii}^n\delta W_i^{n+1} + \sum_{j\in N(i)}M_{ij}^n\delta W_j^{n+1} + \sum_{\tau\in K(i)}\sum_{k,k\neq i}M_{ik}^n\delta W_k^{n+1} = \delta\hat{W}_i$$

The diagonal block is given by :

$$M_{ii}^n = \sigma_i^n\text{Id} + \sum_{j\in N(i)}H_{1,ij}^n + \int_{\partial C_i\cap\partial\Gamma_\infty}\mathcal{A}^+\left(W_i^n, \vec{\eta}\right) + \frac{1}{\text{Re}}\sum_{\tau\in K(i)}\left(\frac{\partial \Upsilon_{\tau,i}^n}{\partial W_i}\right) \tag{25}$$

where $\sigma_i^n = \dfrac{\text{area}(C_i)}{\Delta t^n}$. The extra-diagonal blocks are given by :

$$M_{ij}^n = H_{2,ij}^n \quad , \quad M_{ik}^n = \frac{1}{\text{Re}} \left(\frac{\partial \Upsilon_{\tau,i}^n}{\partial W_k} \right)$$

Note that the linear system is written in « delta » form thus :

$$W^{n+1} = W^n + \delta W^{n+1}$$

4 Domain decomposition for the Navier Stokes equations

In this section we propose a particular implementation of the domain decomposition algorithm formulated in subsection 2.2 in the context of the discretization methods described in section 3. First, we briefly discuss the adopted parallelization strategy and motivate it with regards to the implementation of the domain decomposition algorithm. The next step is to introduce interface unknowns which are here expressed in terms of convective and viscous fluxes. These interface unknowns are used to define a modified formulation of the implicit system (20) in which a distinction is made between purely interior unknowns and interface ones. Finally, we apply a substructuring technique to the resutling system in order to obtain an interface problem whose unknowns are defined in terms of fluxes.

4.1 Parallelization strategy

The parallelization strategy adopted for the single grid flow solver combines domain partitioning techniques and a message-passing programming model. This strategy has been already successfully applied in the single grid case in 2D[12] as well as in 3D[21]. The underlying mesh is assumed to be partitioned into several submeshes, each defining a subdomain. 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 and is well suited for the mixed finite volume/element formulation considered herein. However, mesh partitions with overlapping have a main drawback : they incur redundant floating-point operations. The second possible strategy is based on non-overlapping mesh partitions and incur no more redundant floating-point operations. While updated nodal values are exchanged between the subdomains in overlapping mesh partitions, partially gathered quantities are exchanged between subdomains in non-overlapping ones. ‘ According to the domain decomposition algorithm

formulated in subsection 2.2, 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 viewpoint of the dual discretization of Ω in terms of control surfaces; if Ω_1 and Ω_2 are neighbors then :

$$\Gamma = \Omega_1 \cap \Omega_2 = \bigcup_{C_{1_k} \in \Omega_1, C_{2_k} \in \Omega_2} \partial C_{1_k} \cap \partial C_{2_k}$$

4.2 Domain decomposition algorithm : the discrete case

Here, we discuss the formulation of discrete counterparts of the interface conditions in Eq. (4). To simplify the presentation we consider the case of a decomposition of Ω in two subdomains (see Fig. 2). Based on the mixed finite volume/finite element formulation, the convective and diffusive terms are treated differently by introducing separate interface unknowns.

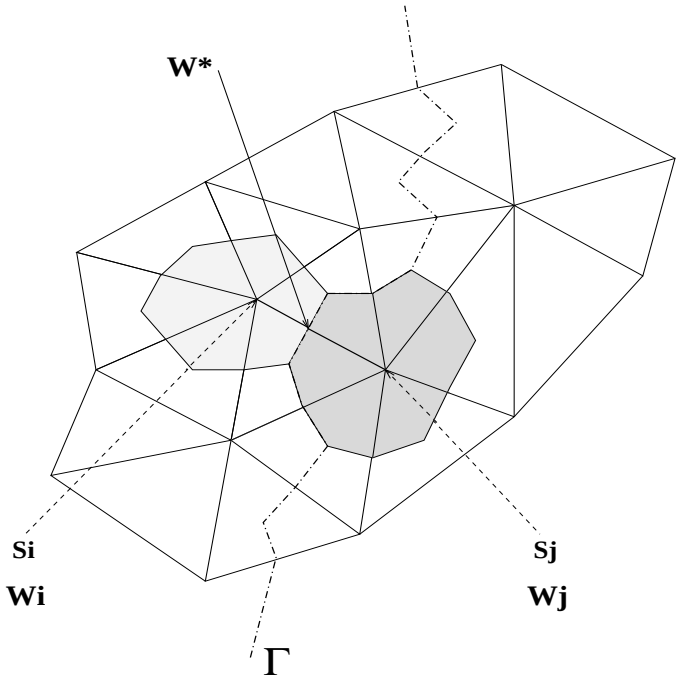

Figure 2: Definition of a redundant variable at an interface $\Gamma = \Omega_1 \cap \Omega_2$

4.2.1 Continuity of 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. The continuity of the normal convective fluxes at the interface $\Gamma = \Omega_1 \cap \Omega_2$ can be written as :

$$\left\{ \begin{array}{rcl} \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i & = & \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j \\ \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j & = & \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i \end{array} \right. \tag{26}$$

where $\tilde{W}^n$ is given by (10) and we have noted $\mathcal{A}^\pm_{\vec{\nu}_{ij}}(\tilde{W}^n) = \mathcal{A}^\pm_R(W_i, W_j, \vec{\nu}_{ij})$. We introduce an auxiliary variable denoted by W^* which is such that :

$$\left\{ \begin{array}{rcl} \left(\mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j\right)|_{s_j} & = & \left(\mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W^*\right)|_{\underline{s_i+s_j}} \\ \left(\mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i\right)|_{s_i} & = & \left(\mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W^*\right)|_{\underline{s_i+s_j}} \end{array} \right. \tag{27}$$

and we define :

$$\Phi_c = |\mathcal{A}_{\vec{\nu}_{ij}}(\tilde{W}^n)|W^* = \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j \tag{28}$$

the associated new unknown of the problem. We can write :

$$\Phi_c = \left(\mathcal{T}(\tilde{W}^n)|\Lambda(\tilde{W}^n)|\mathcal{T}^{-1}(\tilde{W}^n)\right)W^* \quad \Leftrightarrow \quad W^* = \left(\mathcal{T}(\tilde{W}^n)|\Lambda(\tilde{W}^n)|^{-1}\mathcal{T}^{-1}(\tilde{W}^n)\right)\Phi_c$$

The positive and negative parts of this flux are given by :

$$\begin{array}{rcl} \Phi^\pm_c & = & \mathcal{A}^\pm_{\vec{\nu}_{ij}}(\tilde{W}^n)W^* \\ & = & \left(\mathcal{T}(\tilde{W}^n)\Lambda^\pm(\tilde{W}^n)\mathcal{T}^{-1}(\tilde{W}^n)\right)W^* \\ & = & \left(\mathcal{T}(\tilde{W}^n)\Lambda^\pm(\tilde{W}^n)|\Lambda^{-1}(\tilde{W}^n)|\mathcal{T}^{-1}(\tilde{W}^n)\right)\Phi_c \end{array} \tag{29}$$

that can be written in condensed form as :

$$\Phi^\pm_c = P^\pm(\tilde{W}^n)\Phi_c \tag{30}$$

On the other hand, the elementary linearized fluxes associated with the control surfaces C_i and C_j can be written as :

$$\left\{ \begin{array}{rcl} \Phi_c(W_i, W_j, \vec{\nu}_{ij}) & = & \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_i^n) - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) \right) W_i + \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) W_j \\ \Phi_c(W_j, W_i, \vec{\nu}_{ji}) & = & -\mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) W_j - \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_i^n) - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) \right) W_i \end{array} \right.$$

By making the following approximation at the interface :

$$\mathcal{A}_{\vec{\nu}_{ij}}(W_i^n) - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) \cong \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)$$

we can further use the relations (27), (28) and (29) to get the expression of the interface flux using the new variable Φ_c :

$$\left\{ \begin{array}{rcl} \Phi_c^i(W_i, W^\star, \vec{\nu}_{ij}) & = & \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_i^n) - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) \right) W_i + \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) W^* \\ & = & \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_i^n) - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) \right) W_i + P^-(\tilde{W}^n) \Phi_c \\ \Phi_c^j(W^\star, W_j, \vec{\nu}_{ij}) & = & -\mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) W_j - \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n) W^* \\ & = & -\mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) W_j - P^+(\tilde{W}^n) \Phi_c \end{array} \right. \tag{31}$$

Remark. *We could have chosen another linearization of the interface flux instead of (31) in order to retreive the exact quantities expressed in the relations (27), (28) and (29) without any further approximation, but this linerization would have led to a non-conservative flux :*

$$\left\{ \begin{array}{rcl} \Phi_c^i(W_i, W^\star, \vec{\nu}_{ij}) & = & \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_i^n) - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) \right) W_i + \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) W^\star \\ \Phi_c^j(W^\star, W_j, \vec{\nu}_{ij}) & = - & \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_j^n) - \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n) \right) W_j - \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n) W^\star \end{array} \right. \tag{32}$$

On the other hand, there exists a third way of treating the problem, that consists in adopting for the whole subdomain (i.e. internal and interface edges) a different numerical flux function which combines the two previous possibilities. In this approach, the interface flux is conservative and can be expressed directly as a function of the new variable Φ_c :

$$\left\{\begin{array}{lcl} \Phi_c^i(W_i, W^\star, \vec{\nu}_{ij}) & = & \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i + \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W^\star \\ & = & \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i + P^-(\tilde{W}^n)\Phi_c \\ \Phi_c^j(W^\star, W_j, \vec{\nu}_{ij}) & = - & \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j - \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W^\star \\ & = - & \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j - P^+(\tilde{W}^n)\Phi_c \end{array}\right. \tag{33}$$

4.2.2 Continuity of diffusive fluxes

Here, we construct the interface unknowns associated to the viscous fluxes. First of all, we note that the mixed finite element/finite volume formulation adopted for the discretization in space (see subsection 3.1) does not facilitate the implementation of the interface condition for the normal viscous fluxes. 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[29], would have been more appropriate for the realization of this task. Here, in order to facilitate the implementation, we impose the continuity of the whole diffusive flux components in place of the normal ones. As a consequence, for a given interface triangle τ (a triangle whose vertices are situated in the overlapping area), the components of (13) are considered as unknowns. Now, the problem at hand consists in two tasks :

- the construction of 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);
- the definition an associated linearization of the flux components for the implicit time integration.

Using the expression (13) we can write :

$$\begin{array}{rcl} \Upsilon_{\tau,i}(\tau) & = & \text{area}\,(\tau)\left(R(\tau)\dfrac{\partial \phi_i}{\partial x}\mid_\tau + S(\tau)\dfrac{\partial \phi_i}{\partial y}\mid_\tau\right) \\ & = & \text{area}\,(\tau)\left(\begin{pmatrix} 0 \\ r_2(\tau) \\ r_3(\tau) \\ r_4(\tau) \end{pmatrix}\dfrac{\partial \phi_i}{\partial x}\mid_\tau + \begin{pmatrix} 0 \\ r_3(\tau) \\ s_3(\tau) \\ s_4(\tau) \end{pmatrix}\dfrac{\partial \phi_i}{\partial y}\mid_\tau\right) \end{array} \tag{34}$$

where $r_i(\tau)$ and $s_i(\tau)$ are the components of the viscous flux vectors $R(\tau)$ and $S(\tau)$ which are constant on the triangle τ. According to the expressions for the components of $R(W)$ and $S(W)$ (3), we can further write :

$$\Upsilon_{\tau,i}(\tau) = \text{area}\,(\tau) \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \partial_x\phi_i & \partial_y\phi_i & 0 & 0 & 0 \\ 0 & \partial_x\phi_i & 0 & \partial_y\phi_i & 0 \\ 0 & 0 & \partial_x\phi_i & 0 & \partial_y\phi_i \end{pmatrix} \begin{pmatrix} r_2(\tau) \\ r_3(\tau) \\ r_4(\tau) \\ s_3(\tau) \\ s_4(\tau) \end{pmatrix} \tag{35}$$

where $\partial_{x,y}\phi_i = \dfrac{\partial\phi_i}{\partial x, \partial y}$. The above formula can be seen as an alternative linearization of the viscous flux at the interface (with respect to the original one (23)) where the new variables are given by the quantities $r_i(\tau)$ and $s_i(\tau)$. Therefore the coupling between the nodal variables situated 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 (35). Conversely, the coupling between the new interface variables and the nodal variables can be written using the classical linearization :

$$\begin{cases} r_{2,3,4}(\tau) &= \displaystyle\sum_{k_i \in \tau} \frac{\partial r_{2,3,4}(\tau)}{\partial W_{k_i}} W_{k_i} \\ s_{3,4}(\tau) &= \displaystyle\sum_{k_i \in \tau} \frac{\partial s_{3,4}(\tau)}{\partial W_{k_i}} W_{k_i} \end{cases} \tag{36}$$

4.2.3 Formulation of the interface problem

Taking into account Eq. (28), (31), (35) and (36) we can construct an implicit linear system that distinguishes purely interior unknowns (state vectors) from interface ones (normal fluxes). For a two-subdomain decomposition $\Omega = \Omega_1 \cup \Omega_2$ this linear system has the following form :

$$\begin{pmatrix} \mathcal{M}_1 & 0 & \mathcal{M}_{1c} & \mathcal{M}_{1v} \\ 0 & \mathcal{M}_2 & \mathcal{M}_{2c} & \mathcal{M}_{2v} \\ \mathcal{F}_{1c} & \mathcal{F}_{2c} & \text{Id} & 0 \\ \mathcal{F}_{1v} & \mathcal{F}_{2v} & 0 & \text{Id} \end{pmatrix} \begin{pmatrix} W_1 \\ W_2 \\ \Phi_c \\ \Phi_v \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ 0 \\ 0 \end{pmatrix} \tag{37}$$

where $\mathcal{M}_1$ and $\mathcal{M}_2$ are the matrices resulting from the original linearization (23) 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. At this

point, the internal unknowns can be eliminated in favor of the interface ones to yield the following interface system :

$$\begin{array}{rcl} S \begin{pmatrix} \Phi_c \\ \Phi_v \end{pmatrix} & = & \begin{pmatrix} [\mathrm{Id} - (\mathcal{F}_{1c}\mathcal{M}_1^{-1}\mathcal{M}_{1c} + \mathcal{F}_{2c}\mathcal{M}_2^{-1}\mathcal{M}_{2c})]\Phi_c \\ \\ [\mathrm{Id} - (\mathcal{F}_{1v}\mathcal{M}_1^{-1}\mathcal{M}_{1v} + \mathcal{F}_{2v}\mathcal{M}_2^{-1}\mathcal{M}_{2v})]\Phi_v \end{pmatrix} = g \\ g & = & - \begin{pmatrix} [\mathcal{F}_{1c}\mathcal{M}_1^{-1}b_1 + \mathcal{F}_{2c}\mathcal{M}_2^{-1}b_2] \\ \\ [\mathcal{F}_{1v}\mathcal{M}_1^{-1}b_1 + \mathcal{F}_{2v}\mathcal{M}_2^{-1}b_2] \end{pmatrix} \end{array} \tag{38}$$

As usual in this context, once this system has been solved for $(\Phi_c, \Phi_v)^T$, we obtain the values of the purely internal unknowns by performing independent (i.e. parallel) local solves :

$$\left\{ \begin{array}{rcl} W_1 & = & \mathcal{M}_1^{-1}(b_1 - \mathcal{M}_{1c}\Phi_c - \mathcal{M}_{1v}\Phi_v) \\ W_2 & = & \mathcal{M}_2^{-1}(b_2 - \mathcal{M}_{2c}\Phi_c - \mathcal{M}_{2v}\Phi_v) \end{array} \right. \tag{39}$$

In the present study, the interface system (38) is solved using a full GMRES iteration[30].

5 Solution strategy for the local problems

The domain decomposition algorithm proposed in section 4 calls for independent (i.e. parallel) linear system solution steps in each subdomain. Here, we are interested in solving the corresponding linear systems iteratively, the main reasons being that, on one hand, direct solvers are characterized by high memory and CPU requirements and, on the other hand, we would like to study (at least experimentally) the influence of approximate local solves on the convergence of the interface system solver (i.e. GMRES) as well as on the convergence of the overall domain decomposed flow solver. We note in passing that the robustness of Krylov methods such as GMRES, with respect to inexact matrix-vector products has been the subject of recent investigations [2] (see also [3] for a discussion in the context of Schur complement domain decomposition methods).

In this study, a linear multigrid strategy applied at the subdomain level has been adopted for the local solves. The smoother is a pointwise Gauss-Seidel method. More precisely, the multigrid method is used to accelerate the iterative solution of the local linear systems. It is well known that classical relaxation methods such as the Jacobi or Gauss-Seidel methods quickly damp the high frequencies of the error however they do not allow for an efficient treatment of the low frequency components. The basic

idea of the coarse grid correction scheme is to transfer the partially solved solution on a coarser grid in order to transform the low frequencies of the fine grid solution in high frequencies which are then efficiently damped by the standard relaxation methods. The method considered in this study is described in details in [20]-[9]; its main features are the following :

- *Grid coarsening by agglomeration.* The coarsening strategy adopted here is based on the use of macro elements (macro control surfaces) which form the coarse discretizations of the computational domain. In [20] the adopted coarsening algorithm is based on neighboring relations. 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 (e.g. those having a common boundary) to build the macro elements of the coarser level. The main advantage of this method is that it allows for an automatic generation of the coarser discretizations without building any coarse triangulation.

- *Coarse grid approximation for convective terms.* Recall that 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 the finest 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 (see subsection 3.3). Both conservative variables and normal vectors are interpolated between the different grids. The coarse grid variables are deduced by transfer operators. The normal vectors, linked with each coarse mesh macro element, result from the summation of the finer grid vectors (for the fine mesh control surfaces that have a common boundary with the coarse mesh macro element) ; as a result, at most one flux is computed between two macro control volumes.

- *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; we refer to Carré[9] for a more detailed description of the adopted strategy.

- *Inter-grid transfer operators.* A condition to obtain multigrid efficiency is that the summation of the orders of the transfer operators is greater than the order of the partial differential equation to be solved[18]. For instance, this condition, developed in [36] and [19], requires in order to solve the Navier-Stokes equations, that either prolongation or restriction be linear. However, a linear interpolation is not easily built in the agglomeration context. For the solutions of the Euler equations, we keep the same order for both restriction and prolongation which is compatible with the previous condition for the purely convective approximation, and allows building simple and diagonally dominant coarse grid matrices. 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.

6 Numerical results

6.1 Test case definition

The test case under consideration is given by the flow around the `NACA0012` airfoil. Two unstructured triangular meshes have been used whose characteristics are given in Tab. 1 (see Fig. 3 for a partial view of mesh N1).

Table 1: Characteristics of the meshes for the `NACA0012` airfoil

Mesh	# Vertices	# Triangles	# Edges
N3	48792	96896	145688
N4	194480	387584	582064

The following situations have been considered :

S1 : the transonic flow characterized by a Reynolds number of 2000, a free stream Mach number equal to 0.85 and an angle of attack of 0°. The time step is obtained using the rule CFL=$500 \times k_t$ where k_t denotes the time iteration.

S2 : the subsonic flow characterized by a Reynolds number of 73, a free stream Mach number equal to 0.8 and an angle of attack of 10°. The time step is obtained using the rule CFL=$500 \times k_t$ where k_t denotes the time iteration.

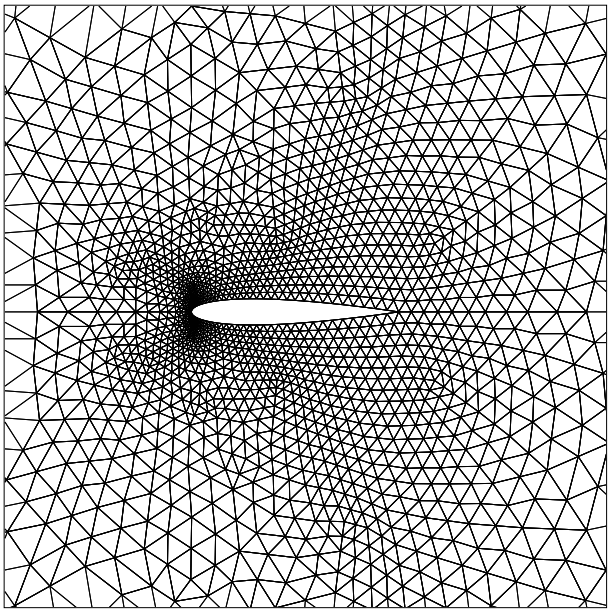

Figure 3: Unstructured triangular mesh around the `NACA0012` airfoil

S3 : the supersonic flow characterized by a Reynolds number of 106, a free stream Mach number equal to 2.0 and an angle of attack of 10°. The time step is obtained using the rule CFL=$50 \times k_t$ where k_t denotes the time iteration.

The results presented below are all characterized by the following points :

- the calculation starts from a uniform flow;
- the local systems induced by the domain decomposition solver are never solved with a high accuracy.

6.2 Computing platforms and conventions

Numerical experiments have been performed on a cluster of 14 `Pentium Pro` computers (dual nodes/500 Mhz with 512 Mb of RAM) (running the LINUX system) interconnected via a 100 Mbit/s `FastEthernet` switch[1]. The MPI implementation is MPICH. The code is written in FORTRAN 77 and the GNU G77 compiler has been used with maximal optimization options.

[1]URL for the description of the platform : http://www-sop.inria.fr/parallel/

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); "# it" is the required number of time steps to reach the steady state (convergence to the steady state is monitored using the normalized energy residual with a non-linear threshold that has been fixed to $\varepsilon_{nl} = 10^{-10}$) ; "Elapsed" denotes the total simultaion time and "CPU" denotes the total CPU time (taken as the maximum value over the local measures); "% CPU" denotes the ratio of "CPU" to "Elapsed" i.e. this ratio gives an idea of the CPU utilization. This ratio is our principal measure of parallel efficiency. The difference between "Elapsed" and "CPU" 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.

6.3 S1 test case

Steady iso-Mach lines for this test case are visualized on Fig. 4. Performance results are given in Tab. 2 and 3 for calculations that have been performed using meshes N3 and N4. In these tables :

- the first part of Tab. 2 and 3 is dedicated to global single grid computations. To be more precise, the original solver is adopted and the global implicit system (20) is approximately solved using Jacobi relaxations. 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 Tab. 2 and 3 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 : the first strategy uses a constant complexity of 3 V-cycles for each local system solution while in the second strategy we impose the (local) linear threshold to $\varepsilon_l = 10^{-1}$. For both cases the linear threshold for the interface system solver (full GMRES) is set to $\varepsilon_i = 10^{-1}$ and the V-cycle is characterized by 2 pre- and 2 post-smoothing steps (the smoother is a pointwise Gauss-Seidel applied at the subdomain level);

- Tab. 3 gives a set of results for calculations performed using mesh N4 and $N_p = 16$;

- in Tab. 2 the parallel speed-up is computed relatively to the elapsed times obtained for $N_p = 4$.

The non-linear convergence to steady-state for the three solution strategies and for $N_p = 24$ are visualized on Fig. 5.

Table 2: S1 test case - timings for the steady state calculation
Global solution strategy (parallel Jacobi linear solver) versus DDM strategy (full GMRES)
Calculations using mesh N3

Method	N_p	N_g	# it	CPU	Elapsed	% CPU	$S(N_p)$
Jacobi ($\varepsilon_g = 10^{-1}$)	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
GMRES ($\varepsilon_i = 10^{-1}$)	4	5	74	1296 sec	1320 sec	98.0	1.0
Local solves : 3 V-cycle(2,2)	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
GMRES ($\varepsilon_i = 10^{-1}$)	4	5	71	1679 sec	1751 sec	96.0	1.0
Local solves : V-cycle(2,2)/$\varepsilon_l = 10^{-1}$	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

This first series of results calls for the following remarks :

- not surprisingly, the domain decomposition solver does outperform the global 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-cycles for the local solves, 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 proposed domain decomposition solver certainly is its parallel efficiency which is here assessed by the " % CPU " ratio.

Table 3: S1 test case - timings for the steady state calculation
Global solution strategy (parallel Jacobi linear solver) versus DDM strategy (full GMRES)
Calculations using mesh N4

Method	N_p	N_g	# it	CPU	Elapsed	% CPU
Jacobi ($\varepsilon_g = 10^{-1}$)	16	1	151	3705 sec	4127 sec	90.0
GMRES ($\varepsilon_i = 10^{-1}$) Local solves : 3 V-cycle(2,2)	16	5	161	3841 sec	3928 sec	98.0
GMRES ($\varepsilon_i = 10^{-1}$) Local solves : V-cycle(2,2)/$\varepsilon_l = 10^{-1}$	16	5	149	8338 sec	8535 sec	97.4

This ratio degrades significantly for the global 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 solves is much more costly than the global solution strategy. On the other hand, when a constant number of V-cycles is used for the local solves, 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 solves is so far dominating the overall simulation time. Despite this fact, on 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 (38) does not make use of any preconditioning technique. In this context, one may ask how such a strategy affects the scalability properties of the domain decomposition solver. A partial answer is illustrated on Fig. 6 and Fig. 7. On these figures, 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 the interface systems are solved with a low accuracy therefore requiring only a few GMRES iterations (6 in average).

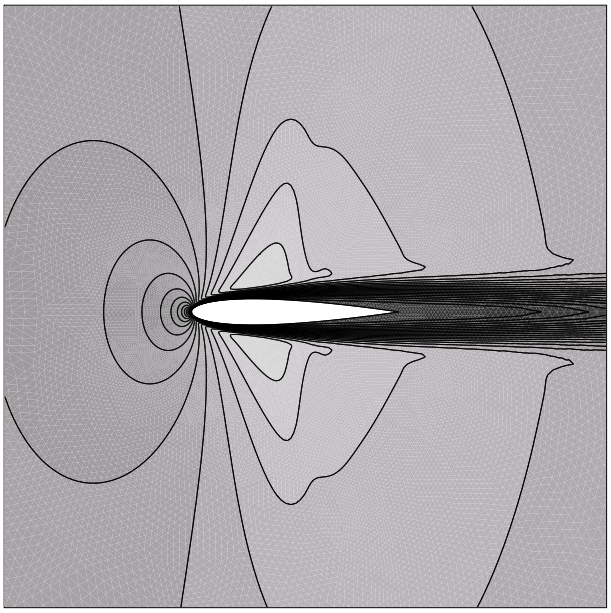

Figure 4: Steady Mach lines for the S1 test case around the NACA0012 airfoil

6.4 S2 test case

Steady iso-Mach lines for this test case are visualized on Fig. 8. Performance results are given in Tab. 4 and 5 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 only have applied a solution strategy based on a fixed number of V-cycles(2,2) for the local solves.

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 time iterations 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 based on more than 3 V-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 $\varepsilon_g = 10^{-1}$ to $\varepsilon_g = 10^{-2}$ allows a faster convergence to steady state at the expense of a 5% increase in the total simulation time.

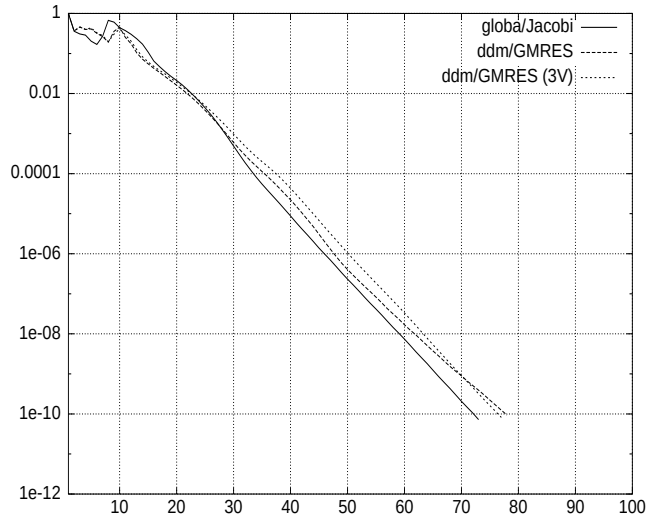

Figure 5: S1 test case - non-linear convergence of the global and the DDM solvers
X-axis : number of time iterations - Y-axis : density residual in log scale

Table 4: S2 test case - timings for the steady state calculation
Global solution strategy (parallel Jacobi linear solver) versus DDM strategy (full GMRES)
Calculations using mesh N3

Method	N_p	N_g	# it	CPU	Elapsed	% CPU	$S(N_p)$
Jacobi ($\varepsilon_g = 10^{-1}$)	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
GMRES ($\varepsilon_i = 10^{-1}$)	4	5	84	1287 sec	1311 sec	98.0	1.0
Local solves : 3 V-cycle(2,2)	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

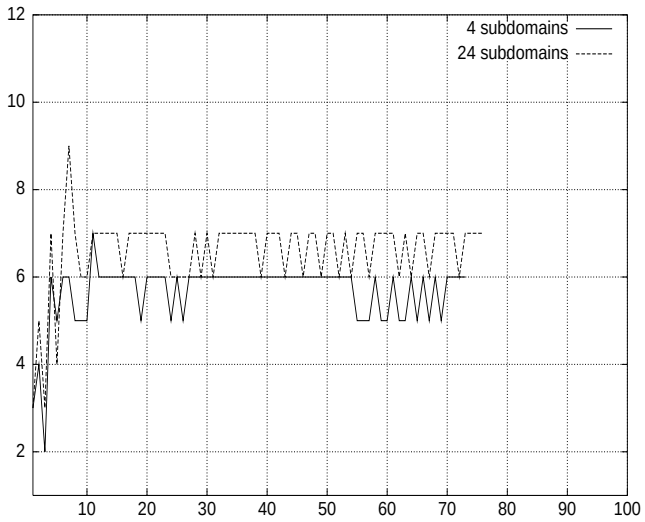

Figure 6: S1 test case - linear convergence of the DDM solver
Interface solver : full GMRES with $\varepsilon_i = 10^{-1}$ - Local solves : 3 V(2,2)-cycles
X-axis : number of time iterations - Y-axis : number of GMRES iterations

Table 5: S2 test case - timings for the steady state calculation
Global solution strategy (parallel Jacobi linear solver) versus DDM strategy (full GMRES)
Calculations using mesh N4

Method	N_p	N_g	# it	CPU	Elapsed	% CPU
Jacobi ($\varepsilon_g = 10^{-1}$)	16	1	125	6211 sec	6903 sec	90.0
Jacobi ($\varepsilon_g = 10^{-2}$)	16	1	117	6527 sec	7255 sec	90.0
GMRES ($\varepsilon_i = 10^{-1}$) Local solves : 3 V-cycle(2,2)	16	5	116	2043 sec	2090 sec	97.5

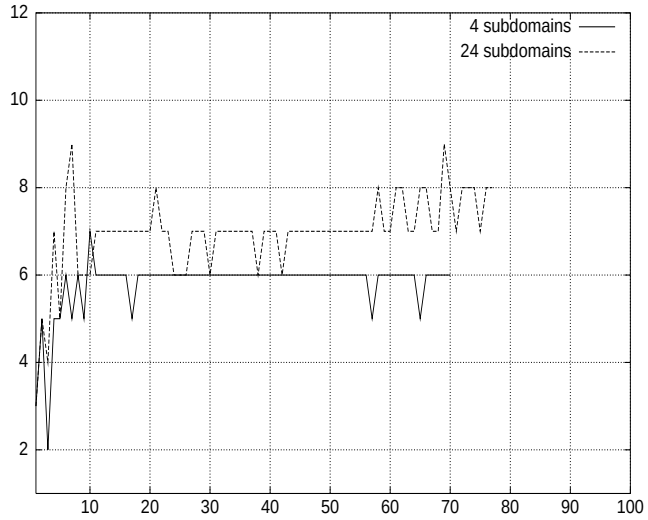

Figure 7: S1 test case - linear convergence of the DDM solver
Interface solver : full GMRES with $\varepsilon_i = 10^{-1}$ - Local solves : V(2,2)-cycles with $\varepsilon_l = 10^{-1}$
X-axis : number of time iterations - Y-axis : number of GMRES iterations

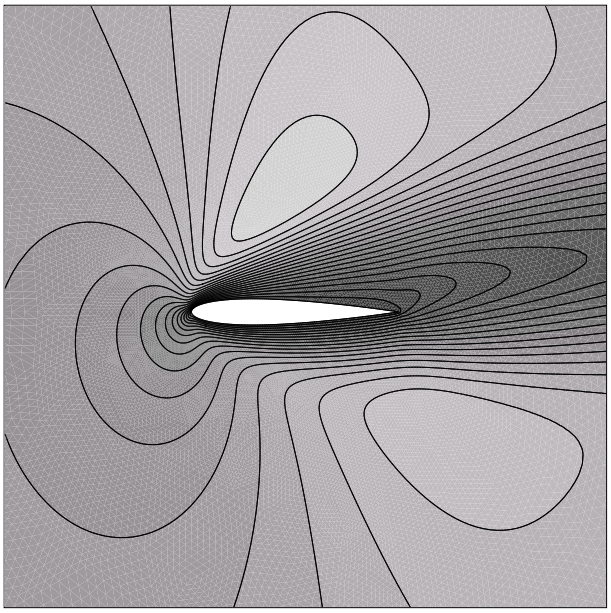

Figure 8: Steady Mach lines for the S2 test case around the NACA0012 airfoil

6.5 S3 test case

Steady iso-Mach lines for this test case are visualized on Fig. 11. Performance results are given in Tab. 6 and 7 for calculations that have been performed using meshes N3 and N4. In these tables :

- the first and the second parts of Tab. 6 and 7 are dedicated to global single grid computations. This time, results are reported for two values of the linear threshold, $\varepsilon_g = 10^{-1}$ and $\varepsilon_g = 10^{-2}$. 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 Tab. 6 and 7 correspond to the application of the domain decomposition method developed in the present study. As with the S1 test case, two strategies have been considered; they differ from the complexity of the local solves : the first strategy uses a constant complexity of 3 V-cycles for each local system solution while in the second strategy we impose the (local) linear threshold to $\varepsilon_l = 10^{-1}$. For both cases the linear threshold for the interface system solver (full GMRES) is set to $\varepsilon_i = 10^{-1}$ and the V-cycle is characterized

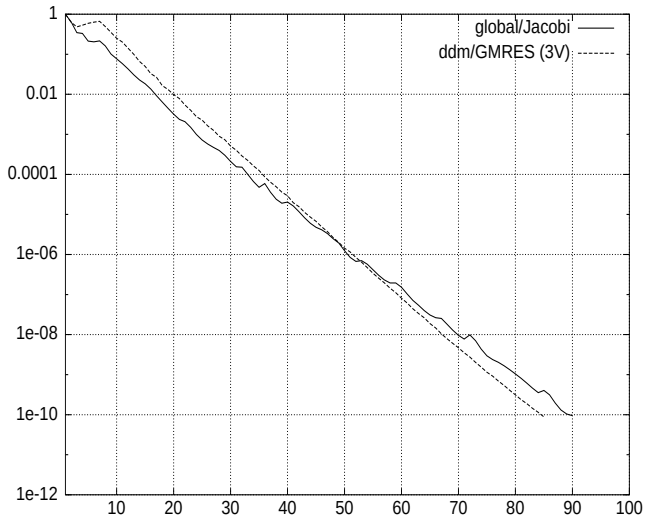

Figure 9: S2 test case - non-linear convergence of the global and the DDM solvers
X-axis : number of time iterations - Y-axis : density residual in log scale

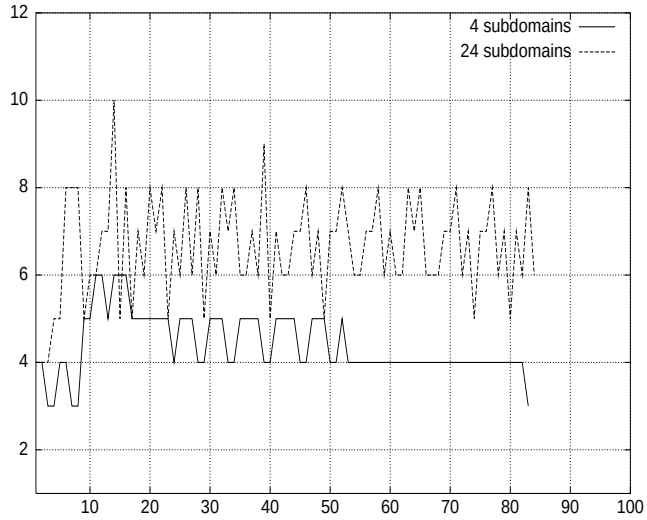

Figure 10: S2 test case - linear convergence of the DDM solver
Interface solver : full GMRES with $\varepsilon_i = 10^{-1}$ - Local solves : 3 V(2,2)-cycles
X-axis : number of time iterations - Y-axis : number of GMRES iterations

by 2 pre- and 2 post-smoothing steps (the smoother is a pointwise Gauss-Seidel applied at the subdomain level);

- Tab. 7 gives a set of results for calculations performed using mesh N4 and $N_p = 16$;
- in Tab. 2 the parallel speed-up 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 visualized on Fig. 12. On this figure, it is clear that the global solution strategy that uses the value $\varepsilon_g = 10^{-1}$ for the linear threshold, is far from yielding the fastest convergence to steady state as compared to the one exhibited by the domain decomposition solver.

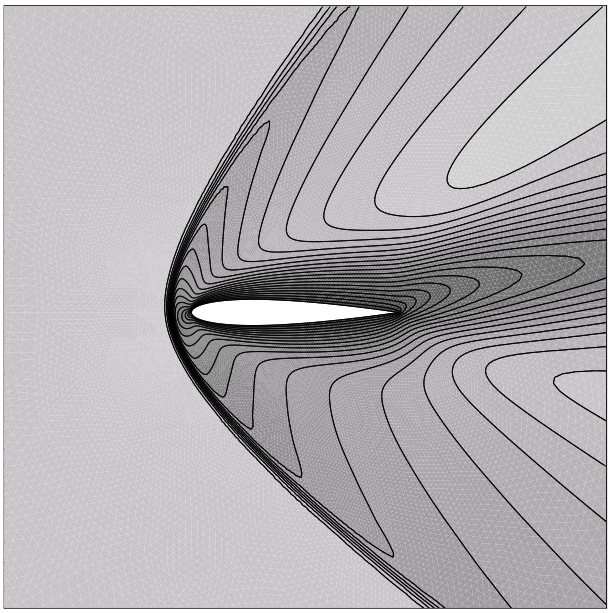

Figure 11: Steady Mach lines for the S3 test case around the `NACA0012` airfoil

The domain decomposition solver based on a constant number of V-cycles for the local solves is the most efficient strategy. 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-cycles for the local solves, shows a 66 % reduction of the

Table 6: S3 test case - timings for the steady state calculation
Global solution strategy (parallel Jacobi linear solver) versus DDM strategy (full GMRES)
Calculations using mesh N3

Method	N_p	N_g	# it	CPU	Elapsed	% CPU	$S(N_p)$
Jacobi ($\varepsilon_g = 10^{-1}$)	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
Jacobi ($\varepsilon_g = 10^{-2}$)	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
GMRES ($\varepsilon_i = 10^{-1}$)	4	5	57	967 sec	979 sec	99.0	1.0
Local solves : 3 V-cycle(2,2)	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
GMRES ($\varepsilon_i = 10^{-1}$)	4	5	57	1163 sec	1202 sec	97.0	1.0
Local solves : V-cycle(2,2)/$\varepsilon_l = 10^{-1}$	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

Table 7: S3 test case - timings for the steady state calculation
Global solution strategy (parallel Jacobi linear solver) versus DDM strategy (full GMRES)
Calculations using mesh N4

Method	N_p	N_g	# it	CPU	Elapsed	% CPU
Jacobi ($\varepsilon_g = 10^{-1}$)	16	1	208	4693 sec	5242 sec	89.5
Jacobi ($\varepsilon_g = 10^{-2}$)	16	1	93	6420 sec	7152 sec	90.0
GMRES ($\varepsilon_i = 10^{-1}$) Local solves : V-cycle(2,2)/$\varepsilon_l = 10^{-1}$	16	5	93	2206 sec	2475 sec	89.0
GMRES ($\varepsilon_i = 10^{-1}$) Local solves : 3 V-cycle(2,2)	16	5	91	1677 sec	1715 sec	98.0

total simulation time for $N_p = 24$ (470 sec for the global solver and 161 sec for the domain decomposition solver). The same comparison for calculations based on mesh N4 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 faster convergence to steady state at the expense of a 36% increase in the total simulation time. Finally, Fig. 13 and Fig. 14 visualize the number of full GMRES iterations at each time step for the domain decomposition solver. It is seen that the required number of GMRES iterations does not increase notably when switching from $N_p = 4$ to $N_p = 24$ subdomains.

7 Conclusion

In this paper, we have described a non-overlapping domain decomposition method for solving the Navier-Stokes equations on unstructured triangular meshes. This method relies on the formulation of an additive Schwarz type algorithm where the interface conditions express the continuity of the normal flux components following an approach already adopted, among others,‘ by Quarteroni and Stolcis[24]. A concrete implementation has been proposed in the context of a mixed finite element/finite volume solver on unstructured triangular meshes. An original aspect of our study consists in the iterative solution of local problems using a multigrid by agglomeration technique. In particular, we have investigated numerically the effect of an approximate solution of local problems on the overall efficiency of the domain decomposition solver. For steady laminar Navier-Stokes flow computations, such a strategy is mandatory to make the domain decomposition solver competitive with classical (global) solution techniques. From this point of view, the proposed domain decomposition solver can

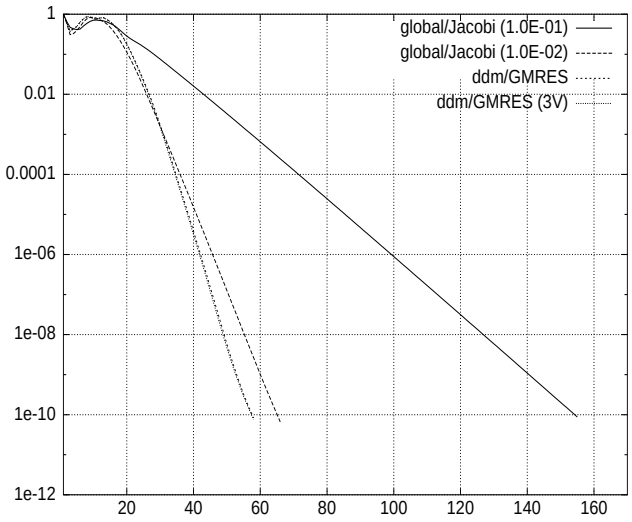

Figure 12: S3 test case - non-linear convergence of the global and the DDM solvers
X-axis : number of time iterations - Y-axis : density residual in log scale

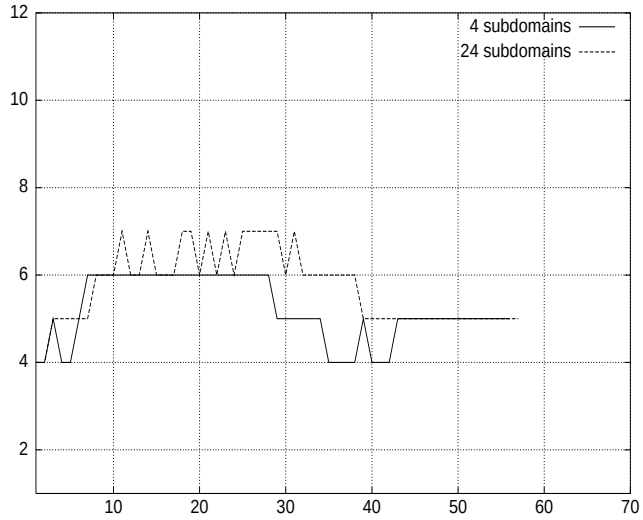

Figure 13: S3 test case - linear convergence of the DDM solver
Interface solver : full GMRES with $\varepsilon_i = 10^{-1}$ - Local solves : 3 V(2,2)-cycles
X-axis : number of time iterations - Y-axis : number of GMRES iterations

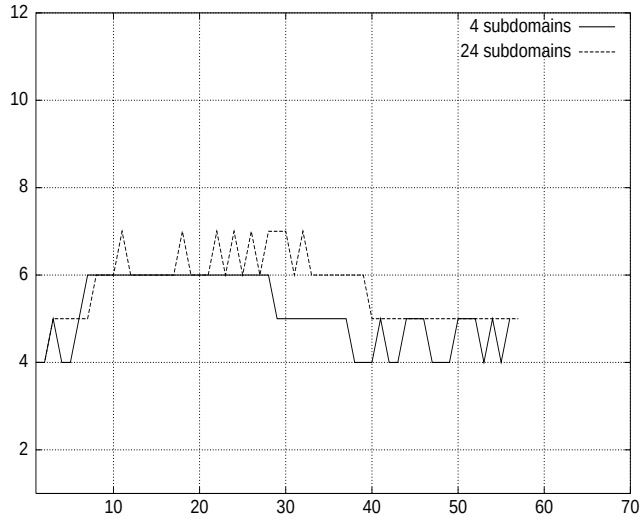

Figure 14: S3 test case - linear convergence of the DDM solver
Interface solver : full GMRES with $\varepsilon_i = 10^{-1}$ - Local solves : V(2,2)-cycles with $\varepsilon_l = 10^{-1}$
X-axis : number of time iterations - Y-axis : number of GMRES iterations

also be viewed as a particular form of additive multigrid in which multigrid acceleration is applied on a subdomain basis, these local calculations being coordinated by an appropriate DDM solver for the interface unknowns. The resulting domain decomposition/multigrid solver has been applied to the computation of several steady flows around a `NACA0012` airfoil. Results have shown that the proposed solver demonstrates higher efficiencies for large number of subdomains as compared to classical global solution strategies.

Ongoing efforts and future works concern the following aspects :

- construction of preconditioners for the interface system (38). We are currently investigating algebraic preconditioning techniques for obtaining approximate inverses to the matrix S;
- assessment of the proposed methodology in the context of the simulation of unsteady viscous flows. In that case, higher order time accuracy is obtained using the defect-correction approach proposed by Martin and Guillard[22];
- extension of the proposed DDM solver to the solution of the Navier-Stokes equations for turbulent flows.

Acknowledgements : the first author acknowledges support from CNES.

References

[1] T.-J. 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, *Proceedings of the 10th Domain Decomposition Methods in Sciences and Engineering*, volume 218 of *Contemporary Mathematics*, pages 23–41. AMS, 1998.

[2] A. Bouras and V. Fraysse. A relaxing strategy for inexact matrix-vector products for krylov methods. Technical Report TR/PA/00/15, CERFACS, 2000.

[3] 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.

[4] 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, ICASE, 1996.

[5] X.-C. Cai, C. Farhat, and M. Sarkis. Schwarz methods for the unsteady compressible Navier-Stokes equations on unstructured meshes. In Z. Shi R. Glowinski, J. Periaux and O. Widlund, editors, *Domain Decomposition Methods in Sciences and Engineering*. John Wiley & Sons, 1997.

[6] 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, *Proceedings of the 10th Domain Decomposition Methods in Sciences and Engineering*, volume 218 of *Contemporary Mathematics*, pages 479–485. AMS, 1998.

[7] X.-C. Cai and O.B. Widlund. Domain decomposition algorithms for indefinite elliptic problems. *SIAM J. Sci. Stat. Comput.*, 13:243–259, 1992.

[8] C. Carlenzoli and A. Quarteroni. Adaptive domain decomposition methods for advection-diffusion problems. In Babuska *et al.*, editor, *Modeling, mesh generation, and adaptive numerical methods for partial differential equations*, volume 75 of *IMA Volumes in Mathematics and its Applications*, pages 169–199. Springer Verlag, 1995.

[9] G. Carré. An implicit multigrid method by agglomeration applied to turbulent flows. *Computers & Fluids*, (26):299–320, 1997.

[10] F. d'Hennezel, P. Le Tallec, and M. Vidrascu. A parallel algorithm for advection-diffusion problem using domain decomposition. Technical Report 33, STPA-ONERA, 1992.

[11] V. Dolean and S. Lanteri. A domain decomposition approach to finite volume solutions of the euler equations on triangular meshes. Technical Report 3751, INRIA, October 1999.

[12] C. Farhat and S. Lanteri. Simulation of compressible viscous flows on a variety of mpps : computational algorithms for unstructured dynamic meshes and performance results. *Comp. Meth. in Appl. Mech. and Eng.*, 119:35–60, 1994.

[13] L. Fezoui and A. Dervieux. Finite element non-oscillatory schemes for compressible flows. In *Eighth France-U.S.S.R.-Italy Joint Symposium on Computational Mathematics and Applications*, number 370 in IAN, Pavie, Italie, 1989.

[14] 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, May 1989.

[15] 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.

[16] F. Gastaldi, L. Gastaldi, and A. Quarteroni. Adaptive domain decomposition methods for advection-dominated equations. *East-West J. Numer. Math.*, 4:165–206, 1996.

[17] F. Gastaldi, L. Gastaldi, and A. Quarteroni. ADN and ARN domain decomposition methods for advection-diffusion equations. In M.-S. Espedal P.-E. Bjorstad and D.-E. Keyes, editors, *Proceedings of the 9th International Conference on Domain Decompositon Methods in Science and Engineering*, pages 334–341. John Wiley & Sons, 1998.

[18] W. Hackbusch. *Multigrid methods and applications*, volume 4 of *Springer series in Computationl Mathematics*. Springer Verlag, 1985.

[19] P.-W Hemker. On the order of prolongations and restrictions in multigrid procedures. *J. Comput. Appl. Math.*, (32):423–429, 1990.

[20] M.-H. Lallemand, Steve S., and Dervieux A. Unstructured multigridding by volume agglomeration : current status. *Computers & Fluids*, 21:397–433, 1992.

[21] S. Lanteri. Parallel solutions of compressible flows using overlapping and non-overlapping mesh partitioning strategies. *Parallel Computing*, 22:943–968, 1996.

[22] R. Martin and V. Guillard. A second order defect correction scheme for unsteady problems. *Computers & Fluids*, 25:9–27, 1996.

[23] M. Paraschivoiu, X.-C. Cai, M. Sarkis, D.-P. Young, and D.-E. Keyes. Multi-domain multi-model formulation for compressible flows: conservative interface coupling and parallel implicit solvers for 3D unstructured meshes. In *Proceedings of the 37th AIAA Aerospace Sciences Meeting and Exhibit*, 1999. AIAA 99-0784.

[24] A. Quarteroni and L. Stolcis. Homogeneous and heterogeneous domain decomposition methods for compressible flow at high reynolds numbers. Technical Report 96/33, CRS4, 1996.

[25] A. Quarteroni and A. Valli. *Theory and application of Steklov-Poincaré operators for boundary value problems*, pages 179–203. Kluwer Academic Publishers, 1991.

[26] A. Quarteroni and A. Valli. Domain decomposition methods for compressible flows. In H. Bulgak and C. Zenger, editors, *Error control and adaptivity in scientific computing*, pages 221–245. Kluwer Academic, 1999.

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

[28] P.L. Roe. Approximate Riemann solvers, parameter vectors and difference schemes. *J. of Comp. Phys.*, 43:357–371, 1981.

[29] P. Rostand and B. Stoufflet. Finite volume Galerkin methods for viscous gas dynamics. Technical Report 863, INRIA, July 1988.

[30] 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.

[31] B. Smith. An optimal domain decomposition preconditionner for the finite element solution of linear elasticity. *SIAM J. Sci. Stat. Comput.*, 13:364–379, 1992.

[32] B. Smith, P. Bjorstad, and W. Gropp. *Domain decomposition and parallel multilevel methods for elliptic partial differential equations*. Cambridge University Press, 1996.

[33] 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.

[34] M.-D. Tidriri. Hybrid newton-krylov domain decomposition methods for compressible flows. In M.-S. Espedal P.-E. Bjorstad and D.-E. Keyes, editors, *Proceedings of the 9th International Conference on Domain Decompositon Methods in Science and Engineering*, pages 532–539. John Wiley & Sons, 1998.

[35] 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.

[36] P. Wesseling. *An introduction to multigrid methods*. John Wiley & Sons, 1991.

[37] 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, *Proceedings of the 10th Domain Decomposition Methods in Sciences and Engineering*, volume 218 of *Contemporary Mathematics*, pages 468–476. AMS, 1998.

[38] J. Xu and X.-C. Cai. A preconditionned gmres method for nonsymmetric or indefinite problems. *Math. Comp.*, 59:311–319, 1992.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399