INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2001; **37**: 625–656 (DOI: 10.1002/fld.184)

A domain decomposition approach to finite volume solutions of the Euler equations on unstructured triangular meshes

Victoria Dolean and Stéphane Lanteri*,[1]

INRIA Sophia Antipolis, Route des Lucioles, Sophia Antipolis Cedex, France

SUMMARY

We report on our recent efforts on the formulation and the evaluation of a domain decomposition algorithm for the parallel solution of two-dimensional compressible inviscid flows. The starting point is a flow solver for the Euler equations, which is based on a mixed finite element/finite volume formulation on unstructured triangular meshes. Time integration of the resulting semi-discrete equations is obtained 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. First, we formulate an additive Schwarz algorithm using appropriate matching conditions at the subdomain interfaces. In accordance with the hyperbolic nature of the Euler equations, these transmission conditions are Dirichlet conditions for the characteristic variables corresponding to incoming waves. Then, we introduce interface operators that allow us to express the domain decomposition algorithm as a Richardson-type iteration on the interface unknowns. 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 (normal) fluxes. Copyright © 2001 John Wiley & Sons, Ltd.

KEY WORDS: domain decomposition method; Euler equations; finite elements; finite volumes; multigrid algorithm; parallel computing; triangular meshes

1. INTRODUCTION

When solving a physical problem modelled by a partial differential equation (PDE), one is generally confronted by a discretization step followed by a sequence of linear system solves. The size or the ill-conditioning of the latter often makes a global or a direct solution approach rather inappropriate. With the advent of parallel computers, domain decomposition algorithms

* Correspondence to: INRIA Sophia Antipolis (Projet Sinus), B.P. 93, 2004 Route des Lucioles, 06902 Sophia Antipolis Cedex, France.
[1] E-mail: Stephane.Lanteri@Inria.fr

Received 22 September 1999
Revised 11 January 2001

have enjoyed an increasing popularity among the scientific community because they define a good framework to derive efficient solvers for the resulting linear systems using the mathematical properties of the initial PDE [1]. As a matter of fact, since the early 1980s, 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 algorithms allow us to solve problems of large size by decomposing them into smaller ones which can be treated by several computers with low memory capacity. In the simplest case, the solution of a subproblem provides the definition of the boundary conditions for the next subproblem when proceeding with a sequential treatement of the different subproblems. The Schwarz algorithm, which illustrates this approach, relies on a spatial decomposition of the initial domain in *overlapping* subdomains. Its original form is often called *multiplicative* since the corresponding iterative operator can be expressed as the product of some local operators related to the local solves. It is obvious that this kind of algorithm is not well suited to parallel architectures. Lately, *additive* variants have been devised, which are characterized by a simultaneous treatment of the subproblems and therefore efficiently exploit parallel computing architectures. To summarize, domain decomposition methods can be classified according to two criteria: *overlapping* versus *non-overlapping* methods according to the spatial decomposition of the global domain, *multiplicative* versus *additive* algorithms according to the interdependence of the local solutions at each iteration. The non-overlapping domain decomposition methods can be of the Schwarz or substructuring (Schur complement or interface system) types, the latest being related to block Gaussian elimination techniques (each block corresponding to a different subdomain). When solving an interface problem, one deals with an operator acting on interface variables, whose discretization is the Schur complement of the global operator [2,3].

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 of the operators involved (or the dominance of the symmetric part of the operators) [4–6]. The situation is less clear for hyperbolic or mixed hyperbolic parabolic models of compressible fluid mechanics. One has to deal with first-order 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 Reference [7], or using a Robin transmission condition and an *iteration by subdomain* algorithm as in References [8–10].

In order to accelerate the convergence of non-overlapping domain decomposition algorithms for the solution of convection–diffusion problems, one can basically consider two directions: the construction of an appropriate preconditioning method for the resulting interface problem or the modification of the interface conditions involved in a Schwarz-type algorithm. For instance, in Reference [11] an optimal Robin–Robin preconditioner is built from local problems with Robin-type conditions at the interface. The second acceleration strategy relies on the notion of absorbing boundary conditions. The absorbing boundary conditions (or artificial boundary conditions) have been introduced for the first time by Engquist and Majda

[12] for the solution of PDEs on an unbounded domain. They are imposed on an artificial boundary such that the solution on the truncated domain is the restriction of the whole solution. Their application to the convection–diffusion or Navier–Stokes equations has been studied by Halpern and Schatzmann in References [13–15]. Recently, Nataf [16] has discovered a similar situation in the framework of domain decomposition methods. Nataf *et al.* [17] have shown that in this context, the use of the absorbing boundary conditions leads to an optimal convergence of a Schwarz-type algorithm. However, these type of boundary conditions involve the use of non-local operators which need to be approximated by partial differential operators. This has been done in the case of a convection–diffusion equation by Japhet [18] using, as an approximation criterion, the minimization of the convergence rate of a Schwarz-type algorithm whose transmission conditions have been optimized.

The objective of the present work is to solve the Euler equations for compressible flows by a non-overlapping domain decomposition method, and more precisely, by a substructuring method. The formulation of a Schwarz-type algorithm for the Euler equations can be found in References [19,20]. In Reference [21] one can find an interface formulation for scalar transport equations by defining a Steklov–Poincaré-type operator. Clerc [19] considered different classes of transmission conditions applied to the solution of linearized and symmetrized hyperbolic systems, such as the Cauchy–Riemann equations. Here, we base our formulation on *classical* transmission conditions that are derived naturally from a weak formulation of the problem. The starting point is a flow solver for the Euler equations, which is based on a mixed 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 [22]. As a result, each pseudo-time step requires the solution of a sparse linear system for the flow variables. In this work, a non-overlapping domain decomposition algorithm is used for advancing the solution at each implicit time step. First, we formulate an additive Schwarz algorithm using appropriate matching conditions at the subdomain interfaces. In accordance with the hyperbolic nature of the Euler equations, these transmission conditions are Dirichlet conditions for the characteristic variables corresponding to incoming waves [20]. Then, we introduce interface operators that allow us to express the domain decomposition algorithm as a Richardson-type iteration on the interface unknowns. 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 (normal) fluxes.

The remaining part of the paper is organized as follows. In Section 2, the Schwarz algorithm and the substructuring technique are introduced in the continuous case for a general linear hyperbolic system. Section 3 describes the characteristics of the starting point mixed finite element/finite volume flow solver for the Euler 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 is generally solved approximately (for example, in the original solver, approximate solutions are obtained using relaxation methods such as the Jacobi or Gauss–Seidel methods). Here, we have adopted the same approach for the local solves induced by the domain decomposition algorithm. In particular, we do not perform direct solution of local problems. In this paper, this strategy is only justified through

numerical experiments as we compare the convergence of the proposed domain decomposition algorithm for completely converged and approximate solutions of the local problems. In order to improve the overall efficiency of the domain decomposition flow solver, the iterative solution of local linear systems based on Jacobi or Gauss–Seidel relaxation methods is accelerated by a linear multigrid strategy by volume agglomeration [23]. This is described in Section 5. In Section 6, the resulting domain decomposition flow solver is evaluated through numerical experiments that are performed on a cluster of PCs interconnected via a 100 Mbit/s FastEthernet switch. Finally, conclusions and future works are presented in Section 7.

2. DOMAIN DECOMPOSITION FOR HYPERBOLIC SYSTEMS

In this section we outline the basic principles for the formulation of a non-overlapping domain decompositon algorithm for hyperbolic systems of PDEs. The proposed framework is greatly inspired from Quarteroni and Valli [3], Nataf [16], Gastaldi *et al.* [9], Quarteroni and Stolcis [20] and Gastaldi and Gastaldi [21]. We also refer to Smith *et al.* [1] and Quarteroni and Valli [2] for a detailed discussion of domain decomposition methods. Most of the discussion here is undertaken in the context of a general linear hyperbolic system. Then, in the next section we naturally extend the proposed ideas to the solution of the Euler equations for compressible flows.

2.1. Hyperbolic systems and boundary conditions

Here, we are interested in the numerical solution of a system of conservation laws of the form

$$\partial_t W + \sum_{i=1}^{d} \partial_{x_i} F_i(W) = 0, \quad W \in \mathbb{R}^p \tag{1}$$

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. However, if W is assumed regular, system (1) can be written in quasi-linear form

$$\partial_t W + \sum_{i=1}^{d} \frac{\partial F_i}{\partial W}(W) \partial_{x_i} W = 0 \quad \text{or} \quad \partial_t W + \sum_{i=1}^{d} A_i(W) \partial_{x_i} W = 0 \tag{2}$$

The $A_i(W) = (\partial F_i / \partial W) W$ are the Jacobian matrices of the flux functions $F_i(W)$ with respect to W. System (1) is said to be hyperbolic if, for any unitary real vector $n \in \mathbb{R}^d$, matrix $\Sigma_{i=1}^{d} A_i(W) n_i$ is diagonalizable with real eingenvalues. We are particularly interested in the situation where system (1) is integrated in time using a backward Euler implicit scheme involving a linearization of the flux functions. In that case we have

$$\frac{\delta W}{\delta t} + \sum_{i=1}^{d} \partial_{x_i} \left[\frac{\partial F_i}{\partial W}(W^n) \delta W \right] = -\operatorname{div}(F(W^n)) \tag{3}$$

where $\delta W = W(x, t^{n+1}) - W(x, t^n) = W^{n+1} - W^n$. When δW is assumed regular, we can write the non-conservative form of system (3)

$$\left[\frac{1}{\delta t} Id + \sum_{i=1}^{d} \partial_{x_i}\left[\frac{\partial F_i}{\partial W}(W^n)\right]\right]\delta W + \sum_{i=1}^{d}\left[\frac{\partial F_i}{\partial W}(W^n)\right]\partial_{x_i}\delta W = -\operatorname{div}(F(W^n)) \tag{4}$$

System (4) can be symmetrized through the multiplication of an operator Σ (see for example Barth [24]) 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:

$$A_0 \delta W + \sum_{i=1}^{d} A_i \partial_{x_i} \delta W = f \tag{5}$$

with

$$\begin{cases} A_0 = \Sigma\left[\dfrac{1}{\delta t} Id + \displaystyle\sum_{i=1}^{d} \partial_{x_i}\left[\frac{\partial F_i}{\partial W}(W^n)\right]\right] \\ A_i = \Sigma\left[\dfrac{\partial F_i}{\partial W}(W^n)\right] \\ f = -\Sigma \operatorname{div}(F(W^n)) \end{cases} \tag{6}$$

Now, let $\boldsymbol{n} = (n_1, \ldots, n_p)$ denote the outward normal vector to $\partial\Omega$; we define $A_{\boldsymbol{n}} W$ as the normal trace of W on $\partial\Omega$, with $A_{\boldsymbol{n}} = \Sigma_{i=1}^{d} A_i n_i$. When dealing with boundary conditions, 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 initial and boundary value problem (IBVP) for system (5). More precisely, the number and type of boundary conditions that must be imposed on $\partial\Omega$ are deduced from the expression of system (5) in terms of characteristic variables and is related to information entering the domain Ω. A more rigourous discussion of boundary conditions treatment for hyperbolic systems from gas dynamics, in terms of characteristic variables, is, for example, given in Reference [25] (see also Quarteroni and Valli [2] for a discussion in the context of domain decomposition algorithms). The operator A_n can be decomposed into positive and negative parts, i.e. $A_{\boldsymbol{n}} = A_{\boldsymbol{n}}^{+} + A_{\boldsymbol{n}}^{-}$. Using the diagonalization of $A_{\boldsymbol{n}} = T\Lambda_{\boldsymbol{n}} T^{-1}$ we have

$$\begin{cases} A_{\boldsymbol{n}}^{\pm} = T\Lambda_{\boldsymbol{n}}^{\pm} T^{-1} \\ \Lambda_{\boldsymbol{n}}^{\pm} = \operatorname{diag}(\lambda_i^{\pm})_{1 \le i \le p} \text{ with } \lambda_i^{\pm} = \dfrac{1}{2}(\lambda_i \pm |\lambda_i|) \\ \text{and with } A_{\boldsymbol{n}}^{+} W = A_{\boldsymbol{n}}^{-} W \end{cases}$$

In order to obtain a well-posed IBVP, we have to impose boundary conditions of the form

$$A_{\boldsymbol{n}}^{-} W = A_{\boldsymbol{n}}^{-} g \tag{7}$$

where $A_{\boldsymbol{n}}^-$ is used to select the information entering the domain Ω.

Then, a well-known result is that the problem

$$\begin{cases} \mathscr{L}W = A_0 W + \sum_{i=1}^{d} A_i \partial_{x_i} W = f, & \text{in } \Omega \\ A_{\boldsymbol{n}}^- W = A_{\boldsymbol{n}}^- g, & \text{on } \partial\Omega \end{cases} \tag{8}$$

with $f \in L^2(\Omega)^p$ and $g \in L_A^2(\partial\Omega)$, has a unique solution $W \in \tilde{H}$ (see, for example, Reference [19]) with

$$\tilde{H} = \left\{ W \in L^2(\Omega)^p \text{ such that } \sum_{i=1}^{d} A_i \partial_{x_i} W \in L^2(\Omega)^p \text{ and } W|_{\partial\Omega} \in L_A^{1/2}(\partial\Omega) \right\}$$

$$\text{with } L_A^{1/2}(\partial\Omega) = \left\{ W \text{ such that } \int_{\partial\Omega} |A_{\boldsymbol{n}}| W \cdot W \, \mathrm{d}\sigma < \infty \right\}$$

This result is used in the next section to formulate a non-overlapping domain decomposition algorithm for the solution of (8).

2.2. Domain decomposition and interface conditions

The domain decomposition approach for solving system (8) 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 $\partial\Omega_i$ that intersects $\partial\Omega$; in addition, appropriate interface conditions have to be added to the definition of the subproblems for the part of $\partial\Omega_i$ that is common to neighbouring subdomains. We shall introduce a non-overlapping domain decomposition algorithm for the following boundary value problem:

$$\mathscr{L}W \equiv A_0 W + \sum_{k=1}^{d} A_k \partial_{x_k} W = f \quad \text{in } \Omega + \text{Boundary conditions on } \partial\Omega \tag{9}$$

Let $\Omega = \cup_{i=1}^{N} \Omega_i$ be a stripwise (for simplicity of presentation) decomposition of Ω and W_i the solution of the local problem

$$\begin{cases} \mathscr{L}W_i = f|_{\Omega_i} = f_i \\ + \text{Boundary conditions on } \partial\Omega \cap \partial\Omega_i \\ + \text{Interface conditions on } \partial\Omega_i \cap \partial\Omega_j \end{cases} \tag{10}$$

Let $\boldsymbol{n_i}$ be the outward normal to $\partial\Omega_i$. The local solution W_i is prolongated by zero on Ω/Ω_i; then a necessary and sufficient condition to insure that $\Sigma_{i=1}^{N} W_i$ is the solution of the global problem (9) is that on $\Gamma = \partial\Omega_i \cap \partial\Omega_j$ the following conditions are verified:

$$A^-_{\boldsymbol{n}_i} W_i + A^+_{\boldsymbol{n}_j} W_j = 0 \quad \text{and} \quad A^+_{\boldsymbol{n}_i} W_i + A^-_{\boldsymbol{n}_j} W_j = 0 \tag{11}$$

This result can be deduced from the variational formulation of problem (9). For simplicity, we consider the two-subdomain case $\Omega = \Omega_1 \cup \Omega_2$. Let $\boldsymbol{n}_{\partial\Omega}$ denote the outward normal on $\partial\Omega$ and $\boldsymbol{n} = \boldsymbol{n}_1 = -\boldsymbol{n}_2$ the outward normal on Γ (directed from Ω_1 to Ω_2), and let $W = W_1 + W_2$ and $f = f_1 + f_2$. We introduce variational formulations of (9)–(10) which are based on the following space of test functions:

$$V = \left\{ X \in L^2(\Omega)^p \text{ such that } \sum_{i=1}^{d} A_i X \in H(\mathrm{div}, \Omega)^p,\ A_{\boldsymbol{n}} X \in L^{1/2}_A(\Gamma) \right\}$$

and we note

$$\mathscr{L}' X \equiv A_0^T X - \sum_{k=1}^{d} \partial_{x_k}(A_k X)$$

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

$$\begin{aligned}
&\iint_\Omega [\mathscr{L} W] X \, \mathrm{d}\omega = \iint_\Omega [\mathscr{L}' X] W \, \mathrm{d}\omega + \int_{\partial\Omega} (A_{\boldsymbol{n}_{\partial\Omega}} W) X \, \mathrm{d}\gamma \\
&= \iint_{\Omega_1} [\mathscr{L}' X] W_1 \, \mathrm{d}\omega + \int_{\partial\Omega_1 \cap \partial\Omega} (A_{\boldsymbol{n}_{\partial\Omega}} W_1) X \, \mathrm{d}\gamma + \iint_{\Omega_2} [\mathscr{L}' X] W_2 \, \mathrm{d}\omega + \int_{\partial\Omega_2 \cap \partial\Omega} (A_{\boldsymbol{n}_{\partial\Omega}} W_2) X \, \mathrm{d}\gamma \\
&= \iint_{\Omega_1} [\mathscr{L} W_1] X \, \mathrm{d}\omega - \int_\Gamma (A_{\boldsymbol{n}_1} W_1) X \, \mathrm{d}\gamma + \iint_{\Omega_2} [\mathscr{L} W_2] X \, \mathrm{d}\omega - \int_\Gamma (A_{\boldsymbol{n}_2} W_2) X \, \mathrm{d}\gamma \\
&= \iint_{\Omega_1} [\mathscr{L} W_1] X \, \mathrm{d}\omega + \iint_{\Omega_2} [\mathscr{L} W_1] X \, \mathrm{d}\omega + \int_\Gamma [A_{\boldsymbol{n}} W_2 - A_{\boldsymbol{n}} W_1] X \, \mathrm{d}\gamma
\end{aligned}$$

If W_1 and W_2 are the (local) solutions of (10) then we must have

$$\int_\Gamma [A_{\boldsymbol{n}} W_1 - A_{\boldsymbol{n}} W_2] X \, \mathrm{d}\gamma = 0 \quad \forall X \in V$$

Therefore, a necessary and sufficient condition to insure that $W_1 + W_2$ is the (global) solution of (9) is

$$A_{\boldsymbol{n}} W_1 = A_{\boldsymbol{n}} W_2 \quad \text{on } \Gamma \tag{12}$$

Since $A_{\boldsymbol{n}}$ is non-singular, then (12) implies that $W_1 = W_2$ on Γ, therefore

$$(T \Lambda^{\pm}_{\boldsymbol{n}} T^{-1}) W_1 = (T \Lambda^{\pm}_{\boldsymbol{n}} T^{-1}) W_2 \quad \text{on } \Gamma \tag{13}$$

which yields

$$A_{\boldsymbol{n}}^{-} W_1 = A_{\boldsymbol{n}}^{-} W_2 \quad \text{and} \quad A_{\boldsymbol{n}}^{+} W_1 = A_{\boldsymbol{n}}^{+} W_2 \tag{14}$$

Conversely, since $A_{\boldsymbol{n}} = A_{\boldsymbol{n}}^{+} + A_{\boldsymbol{n}}^{-}$, conditions (14) imply (12), which concludes the proof.

2.3. A non-overlapping additive Schwarz algorithm

For simplicity of presentation, we assume that the domain Ω is rectangular, $\Omega = [x_a, x_b] \times [y_a, y_b]$, and we consider the case of a non-overlapping decomposition in vertical strips, where the subdomains are defined by $\Omega_i =]\gamma_{i-1}, \gamma_i[\times]y_a, y_b[,\ 2 \le i \le N-1,\ \Omega_1 =]x_a, \gamma_1[\times]y_a, y_b[,$ $\Omega_N =]\gamma_{N-1}, x_b[\times]y_a, y_b[$ (see Figure 1). So the outward normal vectors at the interfaces for the subdomain Ω_i are $n_{i,l} = (-1, 0)$ and $n_{i,r} = (1, 0)$. Consequently, $A_{\boldsymbol{n}_{i,l}}^{-} = -A_{\boldsymbol{n}_{i-1,r}}^{+} = -A^{+}$ and $A_{\boldsymbol{n}_{i-1,r}}^{-} = A^{-}$. We define a Schwarz-type algorithm where the interface transmission conditions are of the form (11); however, according to (7) and (8), we define these interface conditions by selecting the information entering each subdomain. Let $W_i^{(0)}$ denote the initial appoximation of the solution in subdomain Ω_i, then the approximation at the $(k+1)$th iteration (where k defines the iteration of the Schwarz algorithm) is the solution of the problem

$$\begin{cases} \mathscr{L} W_i^{(k+1)} = f & \text{in } \Omega_i \\ A^{+} W_i^{(k+1)} = A^{+} W^{(k)} & \text{on } \Gamma_{i,l} \\ A^{-} W_i^{(k+1)} = A^{-} W_{i+1}^{(k)} & \text{on } \Gamma_{i,r} \\ A_{\boldsymbol{n}}^{-} W_i^{(k+1)} = A_{\boldsymbol{n}}^{-} g & \text{on } \partial\Omega \cup \partial\Omega_i \end{cases} \tag{15}$$

with the convention that $\Gamma_{i,l}$ (respectively $\Gamma_{i,r}$) is the straight line $x = \gamma_{i-1}$ (respectively $x = \gamma_i$). Moreover, we have that $W_i^{(0)} = W_i^n$ and $W_i^{n+1} = W_i^{(K)}$, K being the number of iterations of the above algorithm (n denotes the time step). Clearly, (15) defines an additive Schwarz-type algorithm even though its formulation is somewhat unconventional as it is based on a

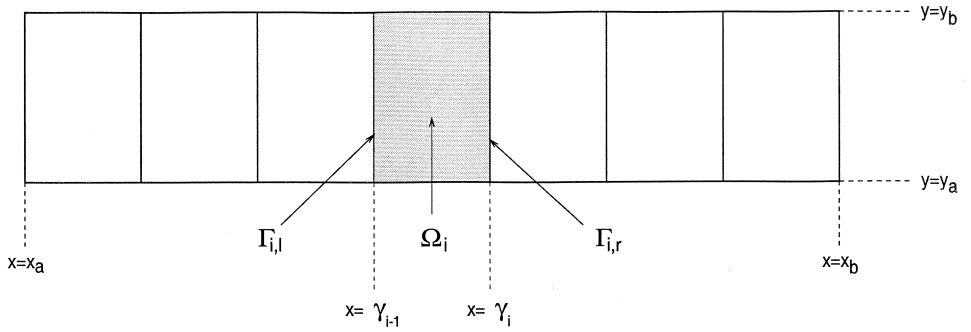

Figure 1. Definition of a vertical strips decomposition.

non-overlapping partitioning of the domain Ω. Such algorithms have been extensively studied by Nataf [16] and Nataf *et al.* [17] for convection–diffusion problems. 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 [12], for improving the convergence of the Schwarz algorithm.

2.4. Substructuring for the definition of an interface problem

In this section we reformulate the previous additive Schwarz algorithm as the iterative solution of an interface system stated in terms of normal fluxes at subdomain interfaces. Adopting the formalism used in Reference [17], we introduce interface operators for each subdomain that take as argument the incoming fluxes at the two interfaces and return outgoing fluxes

$$S^i:\ L_A^{1/2}(\Gamma_{i,l})^p \times L_A^{1/2}(\Gamma_{i,r})^p \times L^2(\Omega_i)^p \times L^2(\Omega_i)^p \to L_A^{-1/2}(\Gamma_{i,l})^p \times L_A^{-1/2}(\Gamma_{i,r})^p$$

such that

$$S^i:\ (\Phi_{i,l}, \Phi_{i,r}, f, g) \to (A^- W_i|_{\Gamma_{i,l}}, A^+ W_i|_{\Gamma_{i,r}}) \tag{16}$$

for $2 \le i \le N-1$, and

$$S^1:\ L_A^{1/2}(\Gamma_{1,r})^p \times L^2(\Omega_i)^p \times L^2(\Omega_i)^p \to L_A^{-1/2}(\Gamma_{1,r})^p$$

$$S^N:\ L_A^{1/2}(\Gamma_{N,l})^p \times L^2(\Omega_i)^p \times L^2(\Omega_i)^p \to L_A^{-1/2}(\Gamma_{N,l})^p$$

with

$$\begin{cases} S^1:\ (\Phi_{1,r}, f, g) \to A^+ W_1|_{\Gamma_{1,r}} \\ S^N:\ (\Phi_{N,l}, f, g) \to A^- W_N|_{\Gamma_{N,l}} \end{cases}$$

for $i=1$ and $i=N$. In the above expressions, $L_A^{-1/2}(\Gamma)$ denotes the dual space of $L_A^{1/2}(\Gamma)$. In each case W_i is the solution of the boundary value problem

$$\begin{cases} \mathscr{L} W_i = f & \text{in } \Omega_i \\ A^+ W_i = \Phi_{i,l} & \text{on } \Gamma_{i,l} \\ A^- W_i = \Phi_{i,r} & \text{on } \Gamma_{i,r} \\ A_{\boldsymbol{n}}^- W_i = A_{\boldsymbol{n}}^- g & \text{on } \partial\Omega \cup \partial\Omega_i \end{cases} \tag{17}$$

The interface operators defined previously are linear and the dependence on their arguments is done via the expressions

$$\begin{cases} A^- W_i|_{\Gamma_{i,l}} = S^i(\Phi_{i,r}, 0, 0, 0)|_{\Gamma_{i,l}} + S^i(0, \Phi_{i,l}, 0, 0)|_{\Gamma_{i,l}} + S^i(0, 0, f, g)|_{\Gamma_{i,l}} \\ A^+ W_i|_{\Gamma_i} = S^i(\Phi_{i,r}, 0, 0, 0)|_{\Gamma_{i,r}} + S^i(0, \Phi_{i,l}, 0, 0)|_{\Gamma_{i,r}} + S^i(0, 0, f, g)|_{\Gamma_{i,r}} \end{cases}$$

In order to simplify the formalism and to emphasize the contribution of each argument, we define the following operators:

$$\begin{cases} S^i_{rr}, S^i_{rl}: \ L_A^{1/2}(\Gamma_{i,l})^p \to L_A^{-1/2}(\Gamma_{i,l})^p \\ S^i_{lr}, S^i_{ll}: \ L_A^{1/2}(\Gamma_{i,r})^p \to L_A^{-1/2}(\Gamma_{i,r})^p \end{cases}$$

such that

$$\begin{cases} S^i_{rr}(\Phi_{i,r}) = S^i(\Phi_{i,r}, 0, 0, 0)|_{\Gamma_{i,l}} \\ S^i_{rl}(\Phi_{i,l}) = S^i(0, \Phi_{i,l}, 0, 0)|_{\Gamma_{i,l}} \\ S^i_{lr}(\Phi_{i,r}) = S^i(\Phi_{i,r}, 0, 0, 0)|_{\Gamma_{i,r}} \\ S^i_{rl}(\Phi_{i,l}) = S^i(0, \Phi_{i,l}, 0, 0)|_{\Gamma_{i,r}} \end{cases}$$

Within this formalism we can now reformulate symbolically the Schwarz-type algorithm in terms of the fluxes $\Phi_{i,l(r)}$ defined above

$$\Phi^{(k+1)} = \mathscr{S}(\Phi^{(k)}) + \mathscr{G} \tag{18}$$

where

$$\Phi = (\Phi_{1,r}, \ldots, \Phi_{N-1,r}, \Phi_{2,l}, \ldots, \Phi_{N,l})^T$$

$$\mathscr{S} = \begin{bmatrix} 0 & S^2_{rr} & \cdots & S^2_{rl} & \cdots & 0 \\ 0 & 0 & S^3_{rr} & \cdots & S^3_{rl} & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & S_N \\ S_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & S^2_{lr} & \cdots & S^2_{ll} & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & \cdots & S^{N-1}_{lr} & \cdots & S^{N-1}_{ll} & 0 \end{bmatrix}$$

$$\mathscr{G} = \begin{bmatrix} S^1(0,f,g)|_{\Gamma_{1,r}} \\ S^2(0,0,f,g)|_{\Gamma_{2,r}} \\ \vdots \\ S^{N-1}(0,0,f,g)|_{\Gamma_{N-1,r}} \\ S^2(0,0,f,g)|_{\Gamma_{2,l}} \\ \vdots \\ S^{N-2}(0,0,f,g)|_{\Gamma_{N-2,l}} \\ S^{N-1}(0,f,g)|_{\Gamma_{N-1,l}} \\ S^N(0,f,g)|_{\Gamma_{N,l}} \end{bmatrix}$$

To summarize, we have shown that the Schwarz algorithm defined by (15) can be interpreted as a relaxation method applied to the system $(I-\mathscr{S})(\Phi)=\mathscr{G}$, which is a Jacobi iteration applied to the interface system (18). As usual, we can accelerate the solution of this problem by applying a Krylov-type method, such as GMRES [26].

3. NUMERICAL SOLUTION OF THE EULER EQUATIONS

In this section we describe the characteristics of the compressible flow solver that is used as a starting point for our study. While doing so, we emphasize the aspects of particular interest to the domain decomposition approach introduced in Section 2, i.e. the upwind finite volume discretization of the convective fluxes and the linearized implicit time integration strategy.

3.1. Mathematical model

Let $\Omega \subset \mathbb{R}^2$ be the computational domain and Γ its boundary. Γ is written 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 Euler equations is given by

$$\frac{\partial W}{\partial t} + \vec{\nabla}\cdot\mathscr{F}(W) = 0, \qquad W = (\rho, \rho\vec{U}, E)^T, \qquad \vec{\nabla} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)^T \tag{19}$$

where $W = W(\vec{x}, t)$; $\vec{x}$ and t respectively denote the spatial and temporal variables while $\mathscr{F}(W) = (F_1(W), F_2(W))^T$ is the conservative flux whose components are given by

$$F_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix}, \qquad F_2(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}$$

In the above expressions, ρ is the density, $\vec{U} = (u, v)^T$ 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

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

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

3.2. Discretization is space

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 neighbouring vertices of s_i by $N(i)$. We associate with 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 Figure 2). 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 $\mathcal{T}_h$.

The spatial discretization method adopted here combines the following elements:

- a finite volume formulation together with upwind schemes for the discretization of the convective fluxes;
- extension to second-order accuracy is obtained by using the Monotonic Upstream Schemes for Conservation Laws (MUSCL) introduced by van Leer [27] and extended to unstructured triangular meshes by Fezoui and Stoufflet [22].

Integrating (19) over C_i and integrating by parts results in

$$\iint_{C_i} \frac{\partial W}{\partial t}\, \mathrm{d}\vec{x} = -\sum_{j \in N(i)} \int_{\partial C_{ij}} \mathcal{F}(W)\cdot\vec{\nu}_i \, \mathrm{d}\sigma \qquad \langle 1 \rangle$$

$$-\int_{\partial C_i \cap \Gamma_w} \mathcal{F}(W)\cdot\vec{n}_i \, \mathrm{d}\sigma - \int_{\partial C_i \cap \Gamma_\infty} \mathcal{F}(W)\cdot\vec{n}_i \, \mathrm{d}\sigma \qquad \langle 2 \rangle \tag{20}$$

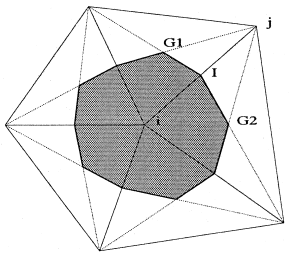

Figure 2. A control surface on a triangular mesh.

where $\partial C_{ij} = \partial C_i \cap \partial C_j$. A first-order finite volume approximation of term $\langle 1 \rangle$ writes as

$$\langle 1 \rangle = W_i^{n+1} - W_i^n + \Delta t \sum_{j \in N(i)} \Phi_{\mathscr{F}}(W_i^n, W_j^n, \vec{v}_{ij}) \tag{21}$$

where $\Phi_{\mathscr{F}}$ denotes a numerical flux function such that

$$\Phi_{\mathscr{F}}(W_i, W_j, \vec{v}_{ij}) \approx \int_{\partial C_{ij}} \mathscr{F}(W) \cdot \vec{v}_i \, \mathrm{d}\sigma, \qquad \vec{v}_{ij} = \int_{\partial C_{ij}} \vec{v}_i \, \mathrm{d}\sigma \tag{22}$$

The numerical flux (22) yields a conservative scheme if for any edge $[s_i, s_j]$, the following condition is verified:

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

Upwinding is introduced in the calculation of (21) by using the approximate Riemann solver of Roe [28], which gives

$$\Phi_{\mathscr{F}}(W_i, W_j, \vec{v}_{ij}) = \frac{\mathscr{F}(W_i) + \mathscr{F}(W_j)}{2} \cdot \vec{v}_{ij} - \left| \mathscr{A}_R(W_i, W_j, \vec{v}_{ij}) \right| \frac{(W_j - W_i)}{2} \tag{23}$$

where

$$\mathscr{A}_R(W_i, W_j, \vec{v}_{ij}) = \left(\frac{\partial \mathscr{F}}{\partial W}(W_i, W_j, \vec{v}_{ij}) \cdot \vec{v} \right)_R$$

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

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

with $\mathscr{F}(W, \vec{v}_{ij}) = \mathscr{F}(W) \cdot \vec{v}_{ij}$. The numerical flux (23) can thus be reformulated as

$$\Phi_{\mathscr{F}}(W_i, W_j, \vec{v}_{ij}) = \mathscr{F}(W_j, \vec{v}_{ij}) - \mathscr{A}^+_{R,ij} \delta W_{ij} \quad \text{or as}$$

$$\Phi_{\mathscr{F}}(W_i, W_j, \vec{v}_{ij}) = \mathscr{F}(W_i, \vec{v}_{ij}) + \mathscr{A}^-_{R,ij} \delta W_{ij}$$

where $\delta W_{ij} = W_j - W_i$. In practice the matrix $\mathscr{A}_R(W_i, W_j, \vec{v}_{ij})$ is computed as $A_{\vec{v}_{ij}}(\tilde{W})$, where $\tilde{W}$ denotes the mean value of Roe [28]. In the general case, the diagonalization of the Jacobian matrix $\mathscr{A}$ is given by

$$\mathscr{A}(W) = \mathscr{T}(W) \Lambda(W) \mathscr{T}^{-1}(W)$$

with

$$\Lambda(W) = \mathrm{diag}(\vec{U} \cdot \vec{v} - c, \vec{U} \cdot \vec{v}, \vec{U} \cdot \vec{v}, \vec{U} \cdot \vec{v} + c)$$

where $c = \sqrt{\gamma(p/\rho)}$ denotes the speed of sound and $\mathscr{T}(W)$ denotes the matrix whose columns are the associated left eigenvectors. The numerical calculation of the convective flux using Equation (23) is first-order accurate in space. The extension to second-order accuracy relies on the MUSCL technique proposed by van Leer [27] and adapted to triangular meshes by Fezoui and Dervieux [29].

3.3. Boundary conditions

Term $\langle 2 \rangle$ in Equation (20) is associated with the boundary conditions of the problem. These are now taken into account in the weak formulation. The following situations are considered:

– *Solid wall*. We impose on Γ_w the slip condition $\vec{U}\cdot\vec{n} = 0$. This condition is introduced in the corresponding term of Equation (20), which results in

$$\int_{\partial C_i \cap \Gamma_w} \mathscr{F}(W)\cdot\vec{n}_i \, \mathrm{d}\sigma = p_i \|\vec{n}\| (0, \tilde{n}_{ix}, \tilde{n}_{iy}, 0)^T \tag{24}$$

– *Far-field boundary*. On Γ_∞, we make use of a uniform flow state vector, i.e. we assume that the flow at infinity is uniform (this assumption is valid for external flows, such as those considered in the results section)

$$\rho_\infty = 1, \quad \vec{U}_\infty = (u_\infty, v_\infty)^T \quad \text{with } \|\vec{U}_\infty\| = 1, \quad p_\infty = \frac{1}{\gamma M_\infty^2} \tag{25}$$

where M_∞ is the far-field Mach number. Here, an *upwind–downwind* flux decomposition is used between the external information (W_∞) and the state vector W_i associated with a vertex $s_i \in \Gamma_\infty$. More precisely, the corresponding boundary integral of term $\langle 2 \rangle$ is evaluated through a non-reflexive version of the Steger and Warming flux decomposition [30]

$$\int_{\partial C_i \cap \Gamma_\infty} \mathscr{F}(W)\cdot\vec{n}_i \, \mathrm{d}\sigma = \mathscr{A}^+(W_i, \vec{n}_{i\infty})\cdot W_i + \mathscr{A}^-(W_i, \vec{n}_{i\infty})\cdot W_\infty \tag{26}$$

3.4. 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 Equation (20)), we obtain the following set of semi-discrete equations:

$$\mathrm{area}(C_i)\frac{\mathrm{d}W_i^n}{\mathrm{d}t} + \Psi(W_i^n) = 0, \quad i = 1, \ldots, N_V \tag{27}$$

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

$$\Psi(W_i^n) = \sum_{j \in N(i)} \Phi_{\mathscr{F}}(W_{ij}, W_{ji}, \vec{v}_{ij}) + \int_{\partial C_i \cap \Gamma} \mathscr{F}(W) \cdot \vec{n}_i \, \mathrm{d}\sigma \tag{28}$$

Explicit time integration procedures for the time integration of Equation (27) are subject to a stability condition expressed in terms of a Courant–Friedrichs–Lewy (CFL) number. An efficient time advancing strategy can be obtained by means of an implicit linearized formulation, such as the one described in Fezoui and Stoufflet [22] and briefly outlined here. First, the implicit variant of Equation (27) writes as

$$\frac{\mathrm{area}(C_i)}{\Delta t^n} \delta W_i^{n+1} + \Psi(W_i^{n+1}) = 0, \quad i = 1, \ldots, N_V \tag{29}$$

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

$$\left(\frac{\mathrm{area}(C_i)}{\Delta t^n} + \frac{\partial \Psi(W^n)}{\partial W} \right) \delta W^{n+1} = -\Psi(W_i^n) \tag{30}$$

In practice we replace the exact Jacobian of the second-order flux $\partial \Psi(W^n)/\partial W$ by an approximate Jacobian matrix $J(W^n)$ resulting from the analytical differentiation of the first-order flux (23)

$$P(W^n) \delta W^{n+1} = \left(\frac{\mathrm{area}(C_i)}{\Delta t^n} + J(W^n) \right) \delta W^{n+1} = -\Psi(W_i^n) \tag{31}$$

The resulting Euler implicit time integration scheme is in fact a modified Newton (see Fezoui and Stoufflet [22] for more details). 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 Equation (30). Moreover, an efficient way to get second-order accurate steady solutions while keeping the interesting properties of the first-order upwind matrix is to use the second-order elementary convective fluxes on the right-hand side of Equation (30). 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.

4. DOMAIN DECOMPOSITION ALGORITHM FOR THE EULER EQUATIONS

In this section we adapt the domain decomposition algorithm proposed in Section 2 to the numerical solution of the Euler equations in the context of the spatial discretization framework described in Section 3. First, we briefly describe the basic parallelization strategy adopted in the original flow solver. Then, we introduce interface unknowns in terms of normal fluxes. The latter are first used to define a modified formulation of the global implicit system (30)

permitting us to distinguish purely interior unknowns from interface ones. Finally, we apply a substructuring technique to this system in order to obtain an interface system whose unknowns are defined in terms of normal 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 tqo dimensions [31] as well as in three dimensions [32]. The underlying mesh is assumed to be partitioned into several submeshes, each defining a subdomain. Basically the same 'old' serial code is executed within every subdomain. Applying this parallelization strategy to the previously described flow solver results in modifications occuring in the main time stepping loop and in the linear solver (which is chosen to be Jacobi in the parallel case) in order to take into account several assembly phases of the subdomain results. The assembly of the subdomain results can be implemented in one or several separated modules and optimized for a given machine. 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. 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. It has been our experience that both the programming effort and the performances are maximized when considering non-overlapping mesh partitions [32]. Here, according to the domain decomposition algorithm formulated in Section 2, it is interesting to consider mesh partitions involving a one-triangle wide overlapping region that is shared by neighbouring subdomains. As a matter of fact, it is easily seen that within this setting, the interface between two neighbouring 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 neighbours 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

In order to be able to construct an interface system of the form (18), 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 3), following a strategy adopted by Clerc [19]. Our approach is, however, different from the one described in Reference [19] in the nature of this redundant variable since, as detailed below, the latter is defined here as the normal flux between two control surfaces belonging to different subdomains. To simplify the presentation we consider the case of a decomposition of Ω in two subdomains. Let $[s_i, s_j]$ be an edge such

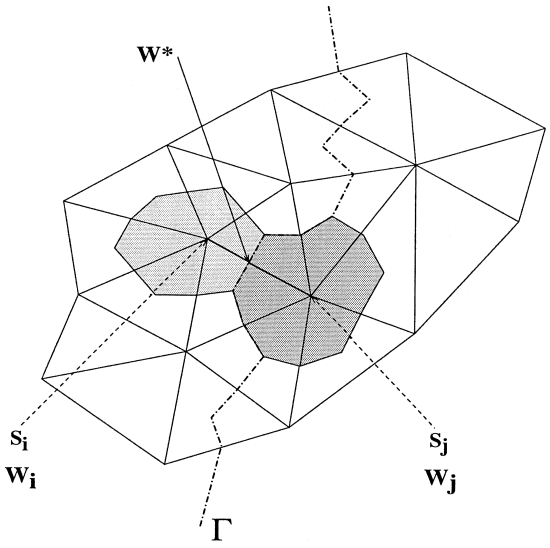

Figure 3. Definition of a redundant variable at an interface $\Gamma = \Omega_1 \cap \Omega_2$.

that C_i (associated with s_i) and C_j (associated with s_j) belong to two neighbouring subdomains. An additive Schwarz formulation is obtained by setting the following interface conditions, which are taken into account in the integral formulation (20) locally in each subdomain:

$$\mathscr{A}^{-}_{\vec{v}_{ij}}(\tilde{W}^n) W_i^{(k+1)} = \mathscr{A}^{-}_{\vec{v}_{ij}}(\tilde{W}^n) W_j^{(k)} \quad \text{and} \quad \mathscr{A}^{+}_{\vec{v}_{ij}}(\tilde{W}^n) W_j^{(k+1)} = \mathscr{A}^{+}_{\vec{v}_{ij}}(\tilde{W}^n) W_i^{(k)} \tag{32}$$

where $\tilde{W}^n$ is the mean value of Roe [28] and where we have noted that $\mathscr{A}^{\pm}_{\vec{v}_{ij}}(\tilde{W}^n) = \mathscr{A}^{\pm}_{R}(W_i, W_j, \vec{v}_{ij})$. The above conditions are expressing the continuity of normal fluxes at the subdomain interface $\Gamma = \Omega_1 \cap \Omega_2$. In the sequel, we simply write W_i instead of $W_i^{(k+1)}$. We introduce an auxiliary variable, denoted by W^*, such that

$$(\mathscr{A}^{-}_{\vec{v}_{ij}}(\tilde{W}^n) W_j)\big|_{s_j} = (\mathscr{A}^{-}_{\vec{v}_{ij}}(\tilde{W}^n) W^*)\big|_{s_{ij}} \quad \text{and} \quad (\mathscr{A}^{+}_{\vec{v}_{ij}}(\tilde{W}^n) W_i)\big|_{s_i} = (\mathscr{A}^{+}_{\vec{v}_{ij}}(\tilde{W}^n) W^*)\big|_{s_{ij}}$$

where $s_{ij} = (s_i + s_j)/2$ and we define

$$\Phi = \big|\mathscr{A}_{\vec{v}_{ij}}(\tilde{W}^n)\big| W^* = \mathscr{A}^{+}_{\vec{v}_{ij}}(\tilde{W}^n) W_i - \mathscr{A}^{+}_{\vec{v}_{ij}}(\tilde{W}^n) W_j \tag{33}$$

the associated new unknown of the problem. We can write

$$\Phi = (\mathscr{T}(\tilde{W}^n)|*(\tilde{W}^n)|\mathscr{T}^{-1}(\tilde{W}^n))W^* \Leftrightarrow W^* = (\mathscr{T}(\tilde{W}^n)|*(\tilde{W}^n)|^{-1}\mathscr{T}^{-1}(\tilde{W}^n))\Phi$$

where $*(\tilde{W}^n)$ is the diagonal matrix whose components are the eigenvalues of $\mathscr{A}_{\vec{v}_{ij}}(\tilde{W}^n)$ and $\mathscr{T}(\tilde{W}^n)$ is the matrix whose columns are the associated left eigenvectors. The positive and negative parts of this flux are given by

$$\begin{aligned}\Phi^{\pm} &= \mathscr{A}^{\pm}_{\vec{v}_{ij}}(\tilde{W}^n)W^* = (\mathscr{T}(\tilde{W}^n)*^{\pm}(\tilde{W}^n)\mathscr{T}^{-1}(\tilde{W}^n))W^* \\ &= (\mathscr{T}(\tilde{W}^n)*^{\pm}(\tilde{W}^n)|*^{-1}(\tilde{W}^n)|\mathscr{T}^{-1}(\tilde{W}^n))\Phi \end{aligned} \tag{34}$$

that we write in condensed form as $\Phi^{\pm} = P^{\pm}(\tilde{W}^n)\Phi$. On the other hand, the elementary fluxes associated with the control surfaces C_i and C_j can be expressed in terms of the auxiliary flux Φ as

$$\begin{cases} \Phi^i(W_i, W^*, \vec{v}_{ij}) = (\mathscr{A}_{\vec{v}_{ij}}(W_i^n) - \mathscr{A}^{-}_{\vec{v}_{ij}}(\tilde{W}^n))W_i + \mathscr{A}^{-}_{\vec{v}_{ij}}(\tilde{W}^n)W^* \\ \qquad\qquad = (\mathscr{A}_{\vec{v}_{ij}}(W_i^n) - \mathscr{A}^{-}_{\vec{v}_{ij}}(\tilde{W}^n))W_i + P^{-}(\tilde{W}^n)\Phi \\ \Phi^j(W^*, W_j, \vec{v}_{ij}) = -(\mathscr{A}_{\vec{v}_{ij}}(W_j^n) - \mathscr{A}^{+}_{\vec{v}_{ij}}(\tilde{W}^n))W_j + \mathscr{A}^{+}_{\vec{v}_{ij}}(\tilde{W}^n)W^* \\ \qquad\qquad = (\mathscr{A}_{\vec{v}_{ij}}(W_j^n) - \mathscr{A}^{-}_{\vec{v}_{ij}}(\tilde{W}^n))W_j + P^{+}(\tilde{W}^n)\Phi \end{cases} \tag{35}$$

Taking into account Equations (33) and (35), we can construct an implicit linear system that distingishes purely interior unknowns (state vectors) from interface ones (normal fluxes)

$$\begin{pmatrix} \mathscr{M}_1 & 0 & \mathscr{M}_{12} \\ 0 & \mathscr{M}_2 & \mathscr{M}_{21} \\ \mathscr{F}_1 & \mathscr{F}_2 & Id \end{pmatrix} \begin{pmatrix} W_1 \\ W_2 \\ \Phi \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ 0 \end{pmatrix} \tag{36}$$

where $\mathscr{M}_1$ (respectively $\mathscr{M}_2$) is the matix that couples the unknowns associated with vertices internal to Ω_1 (respectively Ω_2), whereas $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{M}_{12}$ and $\mathscr{M}_{21}$ are coupling matrices between internal and interface unknowns. These various matrix terms are detailed in Dolean and Lanteri [33]. Now, the internal unknowns can be eliminated in favour of the interface ones to yield the following interface system:

$$S\Phi \equiv [Id - (\mathscr{F}_1\mathscr{M}_1^{-1}\mathscr{M}_{12} + \mathscr{F}_2\mathscr{M}_2^{-1}\mathscr{M}_{21})]\Phi = g \equiv -[\mathscr{F}_1\mathscr{M}_1^{-1}b_1 + \mathscr{F}_2\mathscr{M}_2^{-1}b_2] \tag{37}$$

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

$$W_1 = \mathscr{M}_1^{-1}(b_1 - \mathscr{M}_{12}\Phi) \quad \text{and} \quad W_2 = \mathscr{M}_2^{-1}(b_2 - \mathscr{M}_{21}\Phi) \tag{38}$$

A simple algorithm for solving system (17) is given by the following Richardson-type iteration:

Algorithm 1. *Richardson-type iteration for solving the interface system* $S\Phi = g$.

- Initialization: $\Phi = \Phi^\circ$
- Computation of $g = g_1 + g_2$ (including communication steps to assemble local contributions):
  $g_i = \mathscr{F}_i x_i$, where x_i is obtained through the local resolution: $\mathscr{M}_i x_i = b_i$
- Main parallel loop: $k = 0, \ldots, K$
  - Subdomain Ω_1: $y_1 = \mathscr{M}_{12}\Phi^k$ and $\Phi_1 = \mathscr{F}_1 v_1$, where v_1 is obtained through the local solution: $\mathscr{M}_1 v_1 = y_1$
  - Subdomain Ω_2: $y_2 = \mathscr{M}_{21}\Phi^k$ and $\Phi_2 = \mathscr{F}_2 v_2$, where v_2 is obtained through the local solution: $\mathscr{M}_2 v_2 = y_2$
  - Assembly process (including communication steps): $\Phi^{k+1} = \Phi_1 + \Phi_2 + g$
- If $\|\Phi^{k+1} - \Phi^k\| < \epsilon$ then exit main loop

5. SOLUTION STRATEGY FOR THE LOCAL PROBLEMS

The domain decomposition algorithm proposed in Section 4 calls for independent (parallel) local solves 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 solutions on the convergence of the overall domain decomposed flow solver. In this study, a linear multigrid strategy applied at the subdomain level has been adopted for the local solutions. The smoother is a pointwise Gauss–Seidel method. The method is described in detail in Lallemand *et al.* [23]. Its main features are the following:

- *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, which assembles neighbouring 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.
- *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.

6. NUMERICAL RESULTS

6.1. Test case definition

The test case under consideration is given by the flow around an NACA0012 airfoil. Three unstructured triangular meshes have been used whose characteristics are given in Table I (see Figure 4 for a partial view of mesh N1). Meshes N2 and N3 have been obtained by uniform division of mesh N1. The following situations have been considered:

S1

The subsonic flow at a freestream Mach number equal to 0.3 and an angle of attack of 0°. In this 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

Table I. Characteristics of the meshes around the NACA0012 airfoil.

Mesh	# Vertices	# Triangles	# Edges
N1	3114	6056	9170
N2	12 284	24 224	36 508
N3	48 792	96 896	145 688

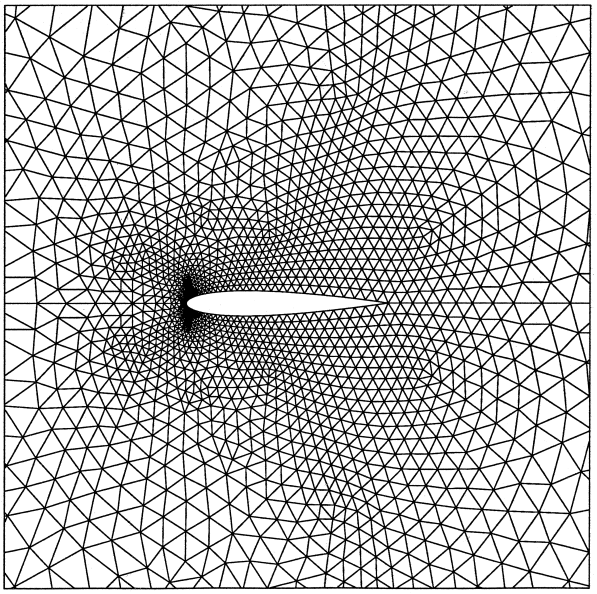

Figure 4. Unstructured triangular mesh around the NACA0012 airfoil.

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 freestream Mach number equal to 0.85 and an angle of attack of 0°. In this case, the time step is obtained using the law $\mathrm{CFL} = 5 \times k_t$, where k_t denotes the time iteration. The Van Albada limiter is used in the MUSCL technique (see Fezoui and Dervieux [29] for more details on this limiter).

6.2. Computing platforms and conventions

Numerical experiments have been performed on a cluster of 12 Pentium Pro 200 MHz computers (running the LINUX system) interconnected via two 100 Mbit/s FastEthernet 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_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; 'Elapsed' denotes the total elapsed execution 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. Interface system solvers

In order to solve the linear system $S\Phi = g$ (see Equation (37)) we have considered the following three strategies:

- a simple Richardson-type iteration (see algorithm 1);
- a full GMRES iteration [26];
- a left preconditioned full GMRES iteration based on a simple algebraic polynomial preconditioner briefly described below. We mention that the objective here is not to devise an appropriate preconditioner for the interface system but simply to have a preliminary evaluation of the role of a preconditioner in the present context.

The expression of the interface matrix of Equation (37) can be written as

$$S = Id - (\mathscr{F}_1 \mathscr{M}_1^{-1} \mathscr{M}_{12} + \mathscr{F}_2 \mathscr{M}_2^{-1} \mathscr{M}_{21}) = Id - A \tag{39}$$

then the construction of the preconditioner starts from the fact that matrix A results from the discretization of a contractant operator [21], therefore, we have that

$$(Id - A)^{-1} = Id + A + A^2 + \cdots \tag{40}$$

An approximate inverse and thus a preconditioner for S is given by

$$S' = Id + A \tag{41}$$

The preconditioned form of (37) writes as

$$S'S\Phi = S'g \Leftrightarrow (Id - A^2)\Phi = g' \tag{42}$$

Clearly the above preconditioner is not computationally cheap since it requires one additional matrix–vector product using matrix A within each GMRES iteration. Therefore, its application will hardly result in a reduction in the execution time; hence, we will mainly assess here the effect of this preconditioner on the required number of GMRES iterations.

6.4. Solution of the first linear system

The following series of numerical experiments concentrate on the solution of the linear system resulting from the first implicit time step, starting from a uniform flow. In this subsection we only consider the subsonic flow test case S1. If not explicitly stated otherwise, the linear thresholds for the local (ε_l) and the interface (ε_i) system solves are given by $\varepsilon_l = \varepsilon_i = 10^{-10}$. We begin with numerical experiments using the value CFL = 20. Table II compares the effective number of iterations for the convergence of the interface system (37) for each mesh of Table

Table II. Convergence of the first linear system (CFL = 20). Subsonic flow test case.

	N_p	N1	N2	N3
Jacobi (global system)	2	322	358	453
Richardson	2	31	33	32
	4	31	32	34
	8	38	38	36
	12	—	—	38
	24	—	—	38
GMRES	2	21	23	24
	4	23	24	24
	8	29	27	25
	12	—	—	28
	24	—	—	28
Preconditioned GMRES	2	11	12	13
	4	12	12	13
	8	15	14	13
	12	—	—	14
	24	—	—	14

I and for various decompositions. Figure 5(left) visualizes the convergence when using the full GMRES iteration and for $N_p = 8$. Timings for mesh N3 and $N_p = 12$ are given in Table III. Several comments can be made

- The convergence of a domain decomposition algorithm is generally assessed with regards to two factors: the number of degrees of freedom, i.e. the characteristic dimension h of the underlying mesh, and the number of subdomains, i.e. the characteristic dimension $H \approx 1/N_p$ of the domain decomposition (each subdomain being viewed as a macro-element). From the results of Table II, it seems that the convergence of the proposed domain decomposition algorithm is weakly sensitive to both characteristic dimensions, at least for the problem stiffness corresponding to the value CFL = 20.
- Following the previous comment, one might legitimately question the influence of the problem stiffness on the convergence of the proposed domain decomposition solver. In the present context, this can be assessed by increasing the CFL number (note that the limit

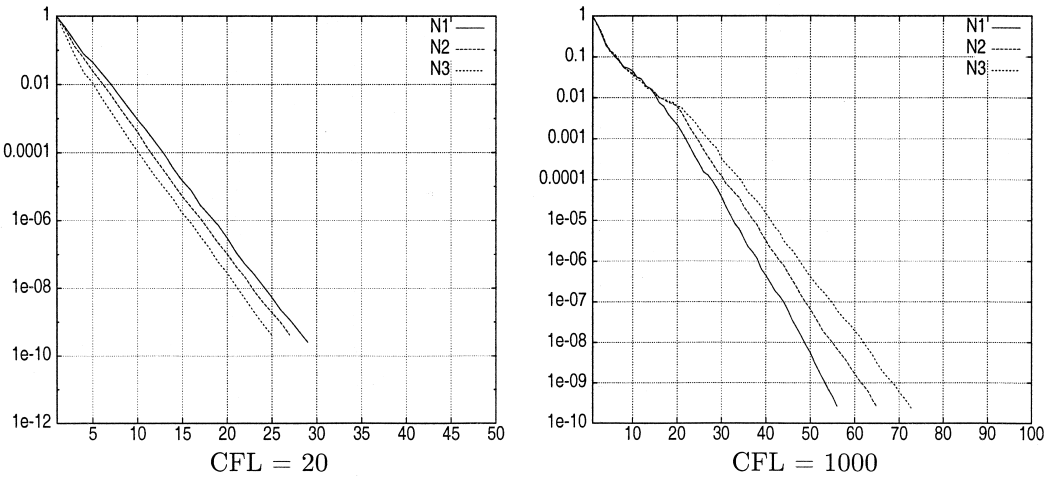

Figure 5. Subsonic flow test case: convergence of the first linear system. Solution of the interface system: full GMRES iteration ($N_p = 8$).

Table III. Timings for the solution of the first linear system (CFL = 20). Subsonic flow test case: mesh N3.

Method	N_p	CPU (s)	Elapsed (s)	% CPU
Jacobi (global system)	12	61	65	94.0
Richardson	12	1058	1074	98.5
GMRES	12	832	846	98.3
Preconditioned GMRES	12	846	862	98.1

should be given by the maximum value that allows a correct calculation of the steady flow, i.e. without incurring negative values of the density or the pressure). Figure 6 compares the convergence of the interface system solution for several values of the CFL number using mesh N3 and for $N_p = 8$, while Figure 5(right) visualizes the convergence for each mesh of Table I using the value CFL = 1000 and for $N_p = 8$; in this case, the number of iterations increases from 56 (mesh N1) to 73 (mesh N3). On the other hand, Figure 7(left) visualizes the influence of the decompositions of mesh N3 for CFL = 1000. The dependence of the convergence on the parameter H is made clearer on this last figure: whereas the number of iterations increased from 24 ($N_p = 2$) to 28 ($N_p = 24$) in the case CFL = 20 (see Table II), it is increasing from 57 ($N_p = 4$) to 98 ($N_p = 24$) for CFL = 1000. The influence of the simple algebraic preconditioning adopted here is depicted in Figure 7(right): the number of iterations is now increasing from 30 ($N_p = 4$) to 52 ($N_p = 24$). The main effect of this preconditioner is to reduce the number of iterations by a factor close to 2 (1.88 for $N_p = 24$). As expected, since the application of the preconditioner basically requires one additional matrix–vector with the original matrix, the gain in the number of iterations does not translate into a gain in execution times;

- It is clear that highly accurate solutions of the local systems result in computational costs that are prohibitively high. Table III shows that in the best case, the domain decomposition approach is about 13 times more expensive than the global solution strategy based on the simple Jacobi solver. From this table it is also seen that the domain decomposition solver demonstrates higher parallel efficiencies as illustrated by the values of the '%CPU' measure;
- There are at least two strategies that can be considered for reducing the overall cost of the domain decomposition solver. The first one consists in adopting a more efficient local

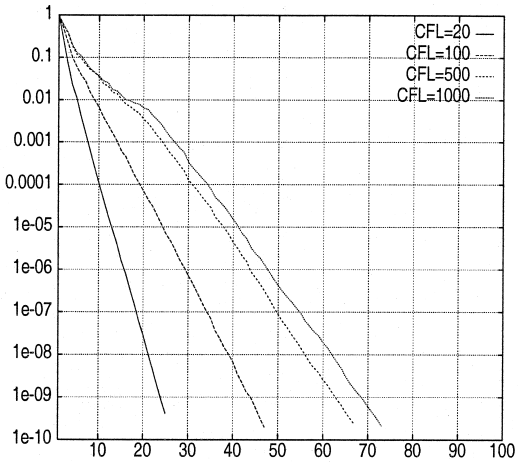

Figure 6. Subsonic flow test case: comvergence of the first linear system (mesh N3). Solution of the interface system: full GMRES iteration ($N_p = 8$). Influence of the CFL number.

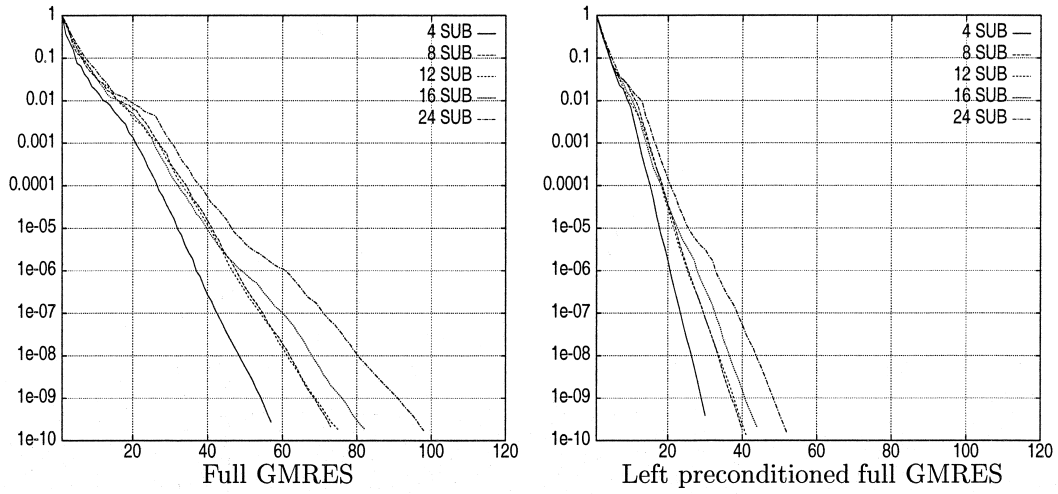

Figure 7. Subsonic flow test case: comvergence of the first linear system. Solution of the interface system (CFL = 1000). Influence of the number of subdomains (mesh N3).

solver. Here this has been achieved by using the multigrid acceleration described in Section 5: a pointwise Gauss–Seidel method has been selected as the smoother in the context of a V-cycle with $\nu_1 = \nu_2 = 2$ and using three coarse grid levels (i.e. $N_g = 4$). Table IV compares the single grid (SG) and the multigrid (MG) local iterative solvers for mesh N3 and $N_p = 8$. Second, one may ask if it is really necessary to solve accurately the subdomain problems. A partial answer is given here in Figure 8, where we compare the convergence of the full GMRES solver for mesh N3, $N_p = 8$ and for two values of the local linear threshold $\varepsilon_l = 10^{-2}$ and $\varepsilon_l = 10^{-10}$. From these figures and the timings in Table IV we note that inexact local solves are not affecting the convergence of the GMRES method and result in a notable reduction in the overall cost of the domain decomposition solver. Of course, the validity of this strategy should be assessed in details in the context of more challenging situations, such as the calculation of unsteady flows.

Table IV. Timings for the solution of the first linear system (CFL = 20). Subsonic flow test case: mesh N3. Comparison of local solution strategies.

Interface system	Local system	N_p	CPU (s)	Elapsed (s)	% CPU
GMRES	SG/$\varepsilon_l = 10^{-10}$	8	1575	1601	98.3
	MG/$\varepsilon_l = 10^{-10}$	8	496	511	97.1
	MG/$\varepsilon_l = 10^{-2}$	8	137	142	96.5

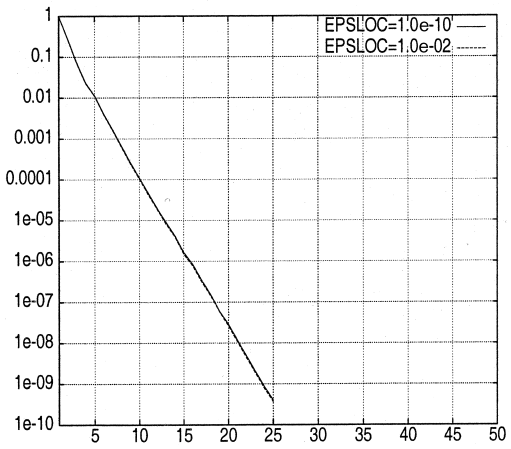

Figure 8. Subsonic flow test case: comvergence of the first linear system. Solution of the interface system: full GMRES iteration ($N_p = 8$). Influence of the accuracy of subdomain solutions (CFL = 20, mesh N3).

6.5. Full steady flow calculations

6.5.1. Subsonic flow test case. Steady isoMach lines for this test case are visualized in Figure 10. The results presented below are all characterized by the following points:

- calculations are performed using mesh N3;
- the CFL number is set to a constant value of 1000;
- the calculation starts from a uniform flow;
- the local systems induced by the domain decomposition solver are never solved with a high accuracy.

One objective of this section is to verify that the last of the above options does not influence the convergence to the steady state compared with the reference convergence of the global implicit approach (see Section 3.4), where the linear system (31) is approximately solved using Jacobi relaxations. Figure 9 visualizes the non-linear convergence in terms of the normalized energy residual for the following situations:

- the global solution strategy where, at each time step, the linear system (31) is solved using Jacobi relaxations with a linear threshold fixed to $\varepsilon_g = 10^{-1}$;
- the proposed DDM strategy where, at each time step, the interface system (37) is solved using full GMRES iterations (without preconditioning) with a linear threshold fixed to $\varepsilon_i = 10^{-1}$; moreover, the local linear systems are solved using either one V-cycle using $v_1 = 4$ pre-smoothing and $v_2 = 4$ post-smoothing steps, or four V-cycles using $v_1 = 2$ pre-smoothing and $v_2 = 2$ post-smoothing steps (in both cases the smoother is a pointwise Gauss–Seidel method) using three coarse grid levels (i.e. $N_g = 4$).

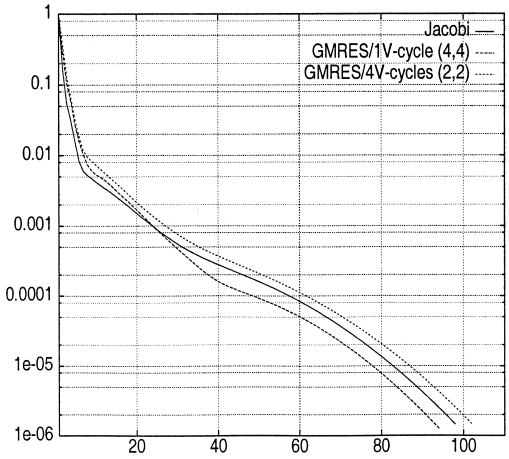

Figure 9. Non-linear convergence (CFL = 1000, mesh N3, $N_p = 8$). Global solution strategy (Jacobi, $\varepsilon_g = 10^{-1}$) versus DDM strategy (full GMRES, $\varepsilon_i = 10^{-1}$).

Effective number of time steps to convergence (initial normalized energy residual reduced by a factor 10^6) as well as execution times are given in Table V. These results call for several comments

- For this particular steady flow test case, the influence of the local solution strategy on the non-linear convergence is rather weak. The reference result is given by the convergence of the global solution strategy based on the Jacobi relaxation method (98 iterations independently of the number of subdomains; for information, increasing the linear threshold from $\varepsilon_g = 10^{-1}$ to $\varepsilon_g = 10^{-2}$ did not result in the reduction of the number of pseudo-time steps to reach the steady state). In comparison, the DDM solver based on the one V-cycle(4, 4) local solution strategy always yields lower numbers of pseudo-time steps with a slight degradation when increasing the number of subdomains.
- Replacing exact subdomain solutions by approximate solutions, for instance, using one V-cycle(4, 4), makes the domain decomposition solver competitive with the global solution strategy. Indeed, we note a 18% reduction of the total execution time for $N_p = 12$ between the global solution strategy and the DDM one. As expected, the DDM solver demonstrates higher parallel efficiencies due to reduced communication overheads.

6.5.2. Transonic flow test case. Steady isoMach lines for this test case are visualized on Figure 10. Here, calculations are stille performed using mesh N3 by the following points; however, this time the CFL number is varying according to the law $\mathrm{CFL} = 5 \times k_t$, where k_t denotes the time iteration. Figure 11 visualizes the non-linear convergence in terms of the normalized energy residual for the following situations:

- the global solution strategy where, at each time step, the linear system (31) is solved using Jacobi relaxations with a linear threshold fixed to $\varepsilon_g = 10^{-1}$ and $\varepsilon_g = 10^{-2}$;

Table V. Subsonic flow test case: timings for the steady state solution. Global solution strategy (Jacobi, $\varepsilon_g = 10^{-1}$) versus DDM strategy (full GMRES, $\varepsilon_i = 10^{-1}$).

Method	N_p	# it	CPU (s)	Elapsed (s)	% CPU
Jacobi (global system)	4	98	2914	2995	97.2
	8	98	1992	2252	88.5
	12	98	1071	1207	88.9
GMRES/4 V-cycle(2, 2)	8	102	3870	4144	93.4
GMRES/1 V-cycle(4, 4)	4	93	2686	2748	97.8
	8	94	1728	1802	95.9
	12	96	911	982	92.7

- the proposed DDM strategy where, at each time step, the interface system (37) is solved using full GMRES iterations (without preconditioning) with a linear threshold fixed to $\varepsilon_i = 10^{-1}$ or $\varepsilon_i = 10^{-2}$; moreover, the local linear systems are solved using one V-cycle using $\nu_1 = 4$ pre-smoothing and $\nu_2 = 4$ post-smoothing steps (the smoother is a pointwise Gauss–Seidel method) using three coarse grid levels (i.e. $N_g = 4$).

Effective number of time steps to convergence (initial normalized energy residual reduced by a factor 10^6), as well as execution times, is given in Table VI. These results call for several comments

- It is clear that setting the linear threshold to $\varepsilon_g = 10^{-1}$ in the global solution strategy is not sufficient to guarantee a correct convergence to steady state. As a matter of fact, the desired level of reduction in the energy residual has not been obtained after 200 pseudo-time steps (this also explains the lower value of the corresponding execution time in Table VI). Consequently, it has been necessary to set the linear threshold to the value $\varepsilon_g = 10^{-2}$.
- On the other hand, switching from $\varepsilon_i = 10^{-1}$ to $\varepsilon_i = 10^{-2}$ in the DDM solution strategy did not improve the convergence to steady state. The required number of pseudo-time steps for $\varepsilon_i = 10^{-1}$ is even lower than what is obtained by setting $\varepsilon_g = 10^{-2}$ in the global solution strategy. This suggests that in the latter case, switching to $\varepsilon_g = 10^{-3}$ would have certainly resulted in a further reduction of the required number of pseudo-time steps; however, at the expense of notably higher execution times.
- By comparing the total execution time of the global solution strategy based on $\varepsilon_g = 10^{-2}$ with that of the DDM solver based on $\varepsilon_i = 10^{-1}$ we conclude that the latter is about three times faster than the former.

7. CONCLUSION AND FUTURE WORKS

In this paper, we have reported on our recent efforts on the design of a non-overlapping domain decomposition algorithm for the solution of two-dimensional compressible inviscid flows. The resulting algorithm has been applied to the computation of a subsonic and a transonic steady flow around an NACA0012 airfoil. An original aspect of our study consists

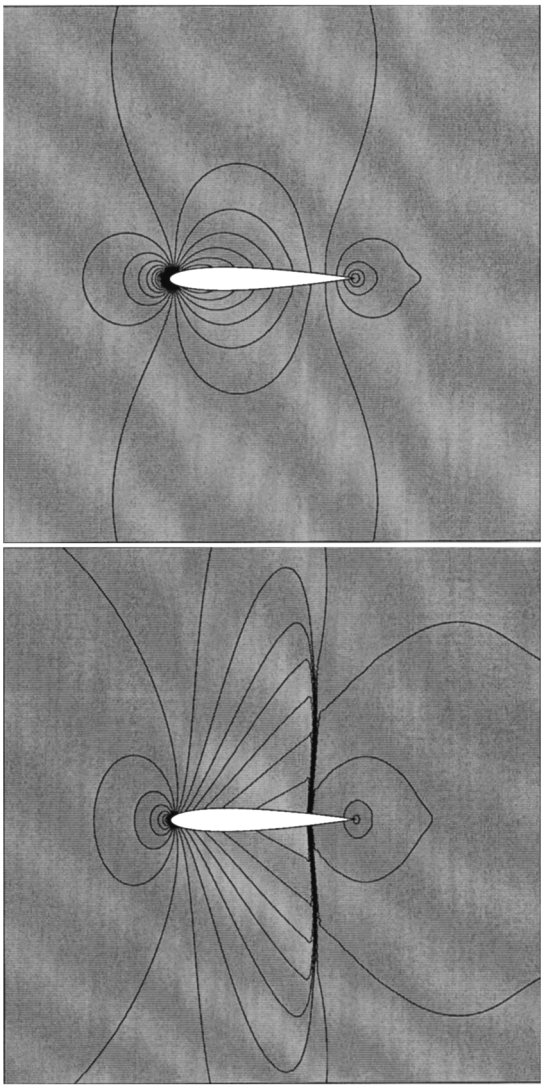

Figure 10. Steady Mach lines for the subsonic (top) and the transonic (bottom) flows.

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 (Euler) 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 also be viewed as a particular form of an additive multigrid

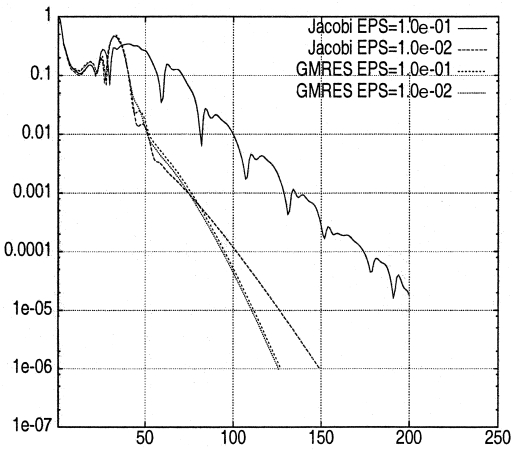

Figure 11. Non-linear convergence (CFL = 1000, mesh N3, $N_p = 8$). Global solution strategy (Jacobi, $\varepsilon_g = 10^{-2}$). DDM strategy (full GMRES, $\varepsilon_i = 10^{-1}$ and $\varepsilon_i = 10^{-2}$).

Table VI. Transonic flow test case: timings for the steady state solution. Global solution strategy (Jacobi, $\varepsilon_g = 10^{-1}$ and $\varepsilon_g = 10^{-2}$). DDM strategy (full GMRES, $\varepsilon_i = 10^{-1}$ and $\varepsilon_i = 10^{-2}$).

Method	N_p	# it	CPU (s)	Elapsed (s)	% CPU
Jacobi (global system), $\varepsilon_g = 10^{-1}$	8	200	1825	2017	90.5
Jacobi (global system), $\varepsilon_g = 10^{-2}$	8	149	4524	5095	88.7
GMRES/1 V-cycle(4, 4), $\varepsilon_i = 10^{-1}$	8	127	1631	1700	96.0
GMRES/1 V-cycle(4, 4), $\varepsilon_i = 10^{-2}$	8	126	2830	2951	95.9

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 successful application of this strategy is clearly illustrated in Figure 11 and Table VI.

Ongoing efforts and future works concern the following aspects:

- convergence analysis of the additive Schwarz algorithm. Our approach to the evaluation of the convergence rate of the additive Schwarz algorithm (15) relies on a Fourier analysis of the linearized two-dimensional Euler equations in the context of a stripwise decomposition of a rectangular domain. Similar approaches have been adopted in References [16,18,34];
- construction of an appropriate preconditioner for the interface system (37). It is interesting to note that the algebraic preconditioner (41) is somewhat of a global nature since it involves the direct application of the matrix A of Equation (39). Other popular preconditioning strategies such as the so-called Neumann–Neumann algorithm (see for instance

Reference [11] for a recent contribution concerning the application of this preconditioner to an advection–diffusion problem) are based on local approximations of the inverse of the interface operator S. As a consequence, it is often necessary to add a coarse (global) space component to the basic preconditioner in order to insure the scalability of the domain decomposition solver. Here, it is not clear that such a coarse space has to be introduced in the definition of the preconditioner (41). On the other hand, as noticed in the Results section, a crude application of this polynomial preconditioner is not efficient. One possible strategy would consist in using an approximate calculation of A in the application of S' using the multigrid by agglomeration method. Note that this is actually done when applying S to a vector; then, the idea would be to used different approximations in the application of S and S' in order to decrease the cost of application of the preconditioner;

- extension of the proposed DDM algorithm to the solution of the Navier–Stokes equations for compressible flows. In that case, the formulation of a non-overlapping domain decomposition algorithm requires the definition of combined convective/diffusive flux interface conditions[20]. The main difficulty that we face is the implicit treatment of such interface conditions in the context of the exisiting flow solver which is based on a mixed finite volume /finite element formulation (upwind schemes for the discretization of convective fluxes/Galerkin approximation of the diffusive fluxes).

ACKNOWLEDGMENTS

Support from CNES is greatly acknowledged.

REFERENCES

1. Smith B, Bjorstad P, Gropp W. *Domain Decomposition and Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press: Cambridge, 1996.
2. Quarteroni A, Valli A. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press: Oxford, 1999.
3. Quarteroni A, Valli A. *Theory and Application of Steklov–Poincaré Operators for Boundary Value Problems*. Applied and Industrial Mathematics, Kluwer Academic Publishers: Dordrecht, 1991; 179–203.
4. Xu J, Cai X-C. A preconditionned GMRES method for nonsymmetric or indefinite problems. *Mathematics of Computing* 1992; **59**: 311–319.
5. Cai X-C, Widlund OB. Domain decomposition algorithms for indefinite elliptic problems. *SIAM Journal of Scientific and Statistical Computing* 1992; **13**: 243–259.
6. Smith B. An optimal domain decomposition preconditionner for the finite element solution of linear elasticity. *SIAM Journal of Scientific and Statistical Computing* 1992; **13**: 364–379.
7. d'Hennezel F, Le Tallec P, Vidrascu M. A Parallel algorithm for advection–diffusion problem using domain decomposition. STPA-ONERA Technical Report No. 33, 1992.
8. Carlenzoli C, Quarteroni A. Adaptive domain decomposition methods for advection–diffusion problems. In *IMA Volumes in Mathematics and its Applications*, *Modeling*, *Mesh Generation*, *and Adaptive Numerical Methods for Partial Differential Equations*, vol. 75. Springer: Berlin, 1995; 169–199.
9. Gastaldi F, Gastaldi L, Quarteroni A. Adaptive domain decomposition methods for advection-dominated equations. *East-West Journal of Numerical Mathematics* 1996; **4**: 165–206.
10. Gastaldi F, Gastaldi L, Quarteroni A. ADN and ARN domain decomposition methods for advection–diffusion equations. In *Proceedings of the 9th International Conference on Domain Decompositon Methods in Science and Engineering*, Bjorstad P-E, Espedal M-S, Keyes D-E (eds). Wiley: New York, 1998; 334–341.
11. Achdou Y, Le Tallec P, Nataf F, Vidrascu M. A domain decomposition preconditionner for an advection–diffusion problem. *Computing Methods in Applied Mechanics and Engineering* 2000; **184**: 145–170.

12. Engquist B, Majda A. Absorbing boundary conditions for the numerical simulation of waves. *Mathematics of Computing* 1977; **31**: 629–651.
13. Halpern L. Artificial boundary conditions for the advection–diffusion equation. *Mathematics of Computing* 1986; **174**: 425–438.
14. Halpern L. Conditions aux limites artificielles pour un système incomplètement parabolique. *Comptes Rendus de l'Académie des Sciences de Paris* 1988; **307**: 413–416.
15. Halpern L, Schatzman M. Artificial boundary conditions for incompressible viscous flows. *SIAM Journal of Matrix Analysis* 1989; **20**: 308–353.
16. Nataf F. On the use of open boundary conditions in block Gauss–Seidel methods for the convection–diffusion equation. Centre de Mathématiques Appliquées, Ecole Polytechnique, Technical Report No. RI284, 1993.
17. Nataf F, Rogier F, Sturler E. Optimal interface conditions for domain decomposition methods. Centre de Mathématiques Appliquées, Ecole Polytechnique, Technical Report No. RI301, 1994.
18. Japhet C. Méthode optimisée d'ordre 2. Application à l'équation d'advection-diffusion. PhD thesis, Université Paris XIII, 1998.
19. Clerc S. Etude de schémas décentrés implicites pour le calcul numérique en mécanique des fluides. Résolution par décomposition de domaine. PhD thesis, Université Paris VI, 1997.
20. Quarteroni A, Stolcis L. Homogeneous and heterogeneous domain decomposition methods for compressible flow at high Reynolds numbers. CRS4 Research Report No. 96/33, 1996.
21. Gastaldi F, Gastaldi L. On a domain decomposition for the transport equation: theory and finite element approximation. *IMA Journal of Numerical Analysis* 1993; **14**: 111–135.
22. Fezoui L, Stoufflet B. A class of implicit upwind schemes for Euler simulations with unstructured meshes. *Journal of Computational Physics* 1989; **84**: 174–206.
23. Lallemand M-H, Steve S, Dervieux A. Unstructured multigridding by volume agglomeration:current status. *Computers & Fluids* 1992; **21**: 397–433.
24. Barth T. *Numerical Methods for Gas Dynamics Systems on Unstructured Meshes*, Lecture Notes in Computational Science and Engineering, vol. 5. Springer: Berlin, 1999; 195–285.
25. Godlewski E, Raviart PA. *Numerical Approximation of Hyperbolic Systems of Conservation Laws*, Applied Mathematical Sciences, vol. 118. Springer: Berlin, 1996.
26. Saad Y, Schultz H. GMRES: generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM Journal of Scientific and Statistical Computing* 1986; **7**: 856–869.
27. van Leer B. Towards the ultimate conservative difference scheme V: a second-order sequel to Godunov's method. *Journal of Computational Physics* 1979; **32**: 361–370.
28. Roe P-L. Approximate Riemann solvers, parameter vectors and difference schemes. *Journal of Computational Physics* 1981; **43**: 357–371.
29. Fezoui L, Dervieux A. Finite element non-oscillatory schemes for compressible flows. In *Proceedings of the Eighth France-U.S.S.R.-Italy Joint Symposium on Computational Mathematics and Applications*. Istituto di Analisi Numerica: Pavia, 1989; Reprint No. 730.
30. Steger J, Warming R-F. Flux vector splitting for the inviscid gas dynamic with applications to finite difference methods. *Journal of Computational Physics* 1981; **40**: 263–293.
31. Farhat C, Lanteri S. Simulation of compressible viscous flows on a variety of MPPs: computational algorithms for unstructured dynamic meshes and performance results. *Computer Methods in Applied Mechanics and Engineering* 1994; **119**: 35–60.
32. Lanteri S. Parallel solutions of compressible flows using overlapping and non-overlapping mesh partitioning strategies. *Parallel Computing* 1996; **22**: 943–968.
33. Dolean V, Lanteri S. A domain decomposition approach to finite volume solutions of the Euler equations on triangular meshes. INRIA Research report No. 3751, 1999.
34. Nataf F, Nier F. Convergence rate of some domain decomposition methods for overlapping and nonoverlapping subdomains. Centre de Mathématiques Appliquées, Ecole Polytechnique, Technical Report No. RI306, 1996.

