ELSEVIER

Comput. Methods Appl. Mech. Engrg. 184 (2000) 427–448

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Parallel linear multigrid algorithms for the acceleration of compressible flow calculations

Gilles Carré *, Luc Fournier, Stéphane Lanteri

INRIA, 2004 route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex, France

Abstract

The interest in unstructured meshes for Computational Fluid Dynamics (CFD) applications appears to be increasingly important in the industrial community. Industrial applications require the numerical simulation of complex flows (i.e. the underlying flows exhibit localized high variations of physical quantities) around or within complex geometries. Unstructured meshes are particularly well suited to these kinds of simulation due to their ability in accurately discretizing complex computational domains and, to their flexibility in dynamically refining and derefining, or deforming, in order to match the underlying flow features. Concerning flow solvers, the main question appears to be the lack of efficiency demonstrated by unstructured mesh solvers compared to structured ones. Many efficient methods developed in the structured context are not easily extensible to unstructured meshes and much research work has yet to be done in this direction. During the last ten years, several such works have demonstrated that multigrid principles can yield robust and efficient unstructured mesh solvers (see for example Lallemand et al. [10], Koobus et al. [9], Carré [1], Mavriplis et al. [16–18]). In this paper, we describe ongoing research activities at INRIA Sophia Antipolis aiming at the construction of efficient and robust unstructured multigrid solvers for complex two-dimensional (2D) and three-dimensional (3D) flow simulations. Both academic and industrial aspects are considered. © 2000 Elsevier Science S.A. All rights reserved.

Keywords: CFD; Navier–Stokes equations; Unstructured meshes; Multigrid methods; Parallelism

1. Introduction

The present work is concerned with the development and the evaluation of parallel linear multigrid algorithms for the acceleration of compressible steady flow calculations. In the first part of this paper, we focus our attention on the two-dimensional (2D) case. The starting point consists in an existing flow solver which is based on an averaged form of the full Navier–Stokes equations coupled to a k–ε turbulence model [1]. The spatial discretization combines finite element and finite volume concepts and is designed on unstructured triangular meshes. Steady state solutions of the resulting semi-discrete equations are obtained by using an Euler implicit time advancing strategy which has the following features: *linearization* (approximate linearization of the convective fluxes and exact differentiation of the viscous terms), *preconditioning* (the Jacobian matrix is based on a first-order Godunov scheme) and *local time stepping and CFL law* (a local time step is computed on each control volume). Each pseudotime step requires the solution of two sparse linear systems (respectively, for the mean flow variables and for the variables associated to the turbulence model). A multigrid strategy by volume agglomeration is introduced in order to allow for an efficient treatment of these two systems. This multigrid method is based on the use of macro elements (macrocontrol volumes) type coarse discretizations of the computational domain where the problem is to be solved on the

[*] Corresponding author.
E-mail address: gilles.carre@inria.fre (G. Carré).

finest mesh. Starting from the finest mesh, a "greedy" type coarsening algorithm is applied to generate automatically the coarse discretizations (see [10]).

Parallelism is introduced in the overall flow solver by using a strategy that combines mesh partitioning techniques and a message passing programming model. The MPI environment is used for the implementation of the required communication steps. This means that both the discrete fluxes calculation and the linear systems solution are performed on a submesh basis; in particular, for the basic linear multigrid algorithm which is multiplicative (i.e. the different levels are treated in sequence with inter-dependencies between the partial results produced on the different levels), this can be viewed as an intra-level parallelization which concentrates on the smoothing steps performed on each member of the grid hierarchy. A necessary and important step in this adaptation was the construction of appropriate data structures for the distribution of coarse grid calculations. Here, this has been achieved by developing a parallel variant of the original "greedy" type coarsening algorithm which now includes additional communication steps for a coherent construction of the communication data structures on the partitioned coarse grids. The constructed parallel linear multigrid algorithm is evaluated in details using two test cases: the laminar viscous flow around a NACA0012 airfoil and the turbulent flow around a RAE8220 profile. Numerical and performance results are presented and discussed for flows calculations that have been performed on a SGI Power Challenge Array MIMD system and a Pentium/P6 200 MHz cluster where the interconnection is realized through a FastEthernet (100 Mbit/s) switch.

The second part of this paper is concerned with the implementation and evaluation of a parallel linear multigrid strategy in the N3S-NATUR industrial CFD package. This activity is supported by a consortium consisting of three end-users of N3S-NATUR: EDF, Snecma and Renault. N3S-NATUR is dedicated to the numerical simulation of complex three-dimensional (3D) compressible steady or transient flows. It is currently able to compute laminar or turbulent flows governed by the Navier–Stokes equations. Turbulence modelling is based on a k–ϵ model using wall laws. It can also handle flows involving one or more different chemical species. The basic spatial approximation methods and time integration strategies are very similar to those described in the previous paragraph. Moreover, N3S-NATUR is able to simulate unsteady flows on deforming meshes (see [21]). The complete software package has been ported on several distributed memory MIMD parallel systems (see [12]). The potentiality of the multigrid approach will be illustrated on several representative test cases.

The remaining of the paper has the following structure: Section 2 is concerned with the mathematical formulation of the problem in the 2D case and its discretization in space and time; Section 3 describes the linear multigrid algorithm; Section 4 is devoted to the parallelization aspects; Section 5 presents and discusses numerical and performance results using two academic test cases around airfoils; Section 6 describes the extension of the present work to the simulation of 3D flows using the N3S-NATUR industrial CFD package; finally, Section 7 draws some conclusion and future works.

2. Mathematical model and approximation methods

2.1. Mathematical model

In this section we concentrate our attention on the 2D case. We consider the complete compressible Navier–Stokes equations coupled with a two-equations turbulence model. When the compressible flow is turbulent, an averaged form of the Navier–Stokes equations is chosen. In that case, the Favre averages are used for the instantaneous variables except for the pressure and the density for which the Reynolds averages are used instead. The Reynolds stress, which appears after the statistical treatment of the equations, is modelled using the Boussinesq assumption; the latter involves an eddy viscosity term defined by a length scale which is related to the dissipation rate (ϵ) and to a velocity scale ($\sqrt{k}$). In order to close the system and determine the turbulent quantities, two additional transport equations, initially described by Launder and Spalding [14] need to be solved. These equations can be written, in a divergence form, as

$$\frac{\partial W}{\partial t}+\frac{\partial F(W)}{\partial x}+\frac{\partial G(W)}{\partial y}=\frac{1}{R_e}\left(\frac{\partial R(W)}{\partial x}+\frac{\partial S(W)}{\partial y}\right)+\frac{\partial\left(\frac{1}{R_t}\tilde{R}(W)\right)}{\partial x}+\frac{\partial\left(\frac{1}{R_t}\tilde{S}(W)\right)}{\partial y}+\Omega(W), \tag{1}$$

where $W(x,y,t)$ is the vector of conservative variables while $F(W)$ and $G(W)$ denote the convective fluxes:

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

with

$$\begin{aligned} &p = (\gamma - 1)\rho e, \\ &E = \rho e + \tfrac{1}{2}\rho(u^2 + v^2) + \rho k, \\ &p' = p + \tfrac{2}{3}\rho k, \\ &E' = E + \beta\rho k \quad \text{with `}\beta = -1 + \tfrac{2}{3(\gamma-1)}. \end{aligned}$$

In the above equations, ρ is the fluid density, u and v are the x and y velocity components; E and e respectively denote the total energy and the internal energy per unit of volume; p is the pressure. The laminar viscous fluxes $R(W)$ and $S(W)$ are given by

$$R(W) = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} + c_x \\ \mu\frac{\partial k}{\partial x} \\ \mu\frac{\partial \epsilon}{\partial x} \end{pmatrix}, \quad S(W) = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} + c_y \\ \mu\frac{\partial k}{\partial y} \\ \mu\frac{\partial \epsilon}{\partial y} \end{pmatrix},$$

where

$$c_x = \frac{\gamma\mu}{P_r}\frac{\partial e}{\partial x} + \beta\mu\frac{\partial k}{\partial x}, \qquad c_y = \frac{\gamma\mu}{P_r}\frac{\partial e}{\partial y} + \beta\mu\frac{\partial k}{\partial y},$$

while $\tilde{R}(W)$ and $\tilde{S}(W)$ stand for the turbulent viscous fluxes:

$$\tilde{R}(W) = \begin{pmatrix} 0 \\ \tau^t_{xx} \\ \tau^t_{xy} \\ u\tau^t_{xx} + v\tau^t_{xy} + \widetilde{c_x} \\ \frac{1}{\sigma_k}\frac{\partial k}{\partial x} \\ \frac{1}{\sigma_\epsilon}\frac{\partial \epsilon}{\partial x} \end{pmatrix}, \quad \tilde{S}(W) = \begin{pmatrix} 0 \\ \tau^t_{xy} \\ \tau^t_{yy} \\ u\tau^t_{xy} + v\tau^t_{yy} + \widetilde{c_y} \\ \frac{1}{\sigma_k}\frac{\partial k}{\partial y} \\ \frac{1}{\sigma_\epsilon}\frac{\partial \epsilon}{\partial y} \end{pmatrix},$$

where

$$\widetilde{c_x} = \frac{\gamma}{P_t}\frac{\partial e}{\partial x} + \frac{1+\beta}{\sigma_k}\frac{\partial k}{\partial x}, \qquad \widetilde{c_y} = \frac{\gamma}{P_t}\frac{\partial e}{\partial y} + \frac{1+\beta}{\sigma_k}\frac{\partial k}{\partial y}.$$

On the other hand

$$\begin{aligned} \tau_{ij} &= \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij} \quad \text{with} \quad u_1 = u \quad \text{and} \quad u_2 = v, \\ \tau^t_{ij} &= \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}. \end{aligned}$$

Finally $\Omega(W)$ is a source term involving the production term P

$$\Omega(W) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \beta(-\rho\epsilon + P) \\ -\rho\epsilon + P \\ c_{\epsilon 1}\frac{\epsilon}{k}P - c_{\epsilon 2}\frac{\rho\epsilon^2}{k} \end{pmatrix},$$

where

$$P = -\left[\tfrac{2}{3}\rho k\delta_{ij} - \mu_t\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3}\frac{\partial u_k}{\partial x_k}\,\delta_{ij}\right)\right]\frac{\partial u_i}{\partial x_j},$$
$$c_\mu = 0.09, \quad c_{\epsilon 1} = 1.44, \quad c_{\epsilon 2} = 1.92, \quad \sigma_\epsilon = 1.3, \quad \sigma_k = 1,$$
$$R_t = \frac{\rho_0 u_0 L_0}{\mu_t}, \quad P_t = \frac{\mu_t c_p}{\lambda_t}, \quad R_e = \frac{\rho_0 u_0 L_0}{\mu_0}.$$

In the above vectors, τ and τ^t, respectively, represent the laminar and turbulent stress tensors, μ is the molecular viscosity and μ_t the turbulent eddy viscosity; R_e and R_t denote the laminar and turbulent Reynolds numbers; P_r and P_t denote the laminar and turbulent Prandtl numbers which are, respectively, equal to 0.72 and 0.9, and γ is the ratio of specific heats of the fluid; μ is computed from the Sutherland law and $\mu_t = c_\mu(\rho k^2/\epsilon)$. The turbulence model is used in conjunction with wall functions [8] deduced from physical considerations. In fact, the previous turbulence equations are only valid for high-Reynolds number regions. The usual strategy is to locate the boundary Γ of the computational domain up to a small distance δ away from the wall in the turbulent region (far-wall zone), which gives two positive Dirichlet conditions for k and ϵ:

$$u_f^2 = \nu\frac{\partial u}{\partial y}_{|\text{wall}} \quad k\,|_\Gamma = \frac{u_f^2}{\sqrt{c_\mu}}, \quad \epsilon\,|_\Gamma = \frac{u_f^3}{k\delta}, \tag{2}$$

where $\nu = \mu/\rho$. Let $\overrightarrow{n}$ and $\overrightarrow{t}$, respectively, denote the normal and tangential vector at any point of the wall Γ. The boundary conditions on the body are expressed from:
- a slip condition, applied to the normal velocity: $\overrightarrow{V}\cdot\overrightarrow{n} = 0$,
- the tangential component of the shear stress which is deduced from the friction velocity $\tau_t = \rho u_f^2$, where

$$\overrightarrow{V}\cdot\overrightarrow{t} = \begin{cases} \frac{R_e\rho\delta u_f^2}{\mu} & \text{if} \quad y^+ = \frac{R_e\delta\rho u_f}{\mu} < 11.6, \\ u_f\left[\frac{1}{\kappa}\log\left(\frac{R_e\rho\delta u_f}{\mu}\right) + C\right] & \text{otherwise,} \end{cases}$$

where the constants κ and C are set to $\kappa = 0.419$ and $C = 5.445$.

2.2. Spatial discretization methods

The spatial approximation is a combination of both finite volume and finite element methods. The computational domain Ω is assumed to be a bounded polygon and we introduce the following definitions:
- τ_h is a triangulation of Ω,
- n_h is the total number of vertices in τ_h,
- ϕ_i^T is the P1 finite element basis function associated with the vertex s_i of triangle T.

We derive a new finite volume partition of Ω, called the dual mesh of τ_h and made of the control volumes C_i around each vertex s_i (see Fig. 1).

The spatial discretization of Eq. (1) is derived from the combination of a finite volume scheme applied to the convective terms and a finite element scheme applied to the diffusive and source terms. Theoretical proofs of the compatibility between these two formulations can be found in [19]. The semi-discrete system, resulting from the application of the Green–Ostrogradsky's formula, can be written as

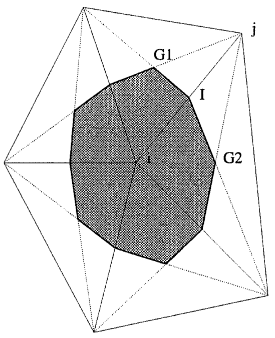

Fig. 1. Control volume C_i.

$$\operatorname{area}(C_i)\frac{\delta W_i^{n+1}}{\Delta t_i}+\int_{\partial C_i}(F(W)\eta_x+G(W)\eta_y)\,\mathrm{d}\sigma=-\frac{1}{R_e}\sum_{T,s_i\in T}\int\int_T\left(R(W)\frac{\partial\phi_i^T}{\partial x}+S(W)\frac{\partial\phi_i^T}{\partial y}\right)\mathrm{d}\vec{x}$$
$$-\sum_{T,s_i\in T}\int\int_T\frac{1}{R_t}\left(\tilde{R}(W)\frac{\partial\phi_i^T}{\partial x}+\tilde{S}(W)\frac{\partial\phi_i^T}{\partial y}\right)\mathrm{d}\vec{x}$$
$$+\sum_{T,s_i\in T}\int\int_T\Omega(W)\phi_i^T\,\mathrm{d}\vec{x},\tag{3}$$

where $\delta W_i^{n+1}=W_i^{n+1}-W_i^n$. Solving the discrete system consists in finding, at each new time level t^{n+1}, the vector W_i^{n+1} using Eq. (3) where the following approximation methods are used:

- *Diffusive and source terms* are calculated on each triangle using a classical P1-Galerkin (centered) formulation.
- *Convective terms*: The flux between two neighboring control volumes C_i and C_j is calculated through the interface ∂C_{ij} (see Fig. 2) using an upwind finite volume formulation. The contribution for mesh vertex s_i is obtained by summing the elementary fluxes calculated with all neighbors s_j:

$$\int_{\partial C_i}\tilde{F}(W,\vec{\eta})\,\mathrm{d}\sigma=\sum_{s_j\in K(s_i)}\int_{\partial C_{ij}}\tilde{F}(W,\vec{n})\,\mathrm{d}\sigma+\int_{\partial C_i\cap\Gamma_h}\tilde{F}(W,\vec{\eta})\,\mathrm{d}\sigma,\tag{4}$$

where

$\tilde{F}(W,\vec{\eta})=F(W)\eta_x+G(W)\eta_y$,
$\Gamma_h=\partial\Omega$ corresponds to solid and farfield boundaries,
$\partial C_{ij}=\partial C_i\cap\partial C_j=[G_{1,ij},I_{ij}]\cup[I_{ij},G_{2,ij}]$,
$K(s_i)$ is the set of neighboring vertices to s_i, and
$\vec{n}$ is the unit normal vector to the interface ∂C_{ij}.

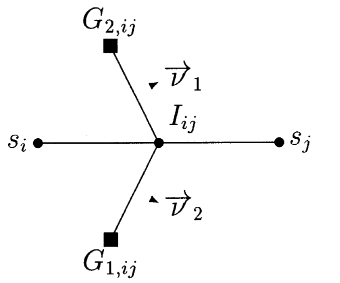

Fig. 2. Interface ∂C_{ij} separating vertices s_i and s_j.

For the mean flow variables, the flux through the ∂C_{ij} interface is calculated using Roe's [22] numerical flux function:

$$\int_{\partial C_{ij}} \tilde{F}(W, \vec{n})\, \mathrm{d}\sigma = \Phi^{\mathrm{Roe}}(W_i, W_j, \vec{v}_{ij}),$$

where $\vec{v_{ij}} = \int_{\partial C_{ij}} \vec{n}\, \mathrm{d}\sigma$ is the normal vector to the interface ∂C_{ij}. The flux developed in [13] is used for the turbulent variables. This flux has first been introduced for solving chemically reactive Riemann problems and has proved to preserve the positivity of each component. Finally, the Steger and Warming [23] numerical flux function is used at the farfield boundaries; to be more precise, upwinding is used in conjunction with a given uniform state W_∞ which is supposed to be representative of the flow at these boundaries.

The above numerical flux integration is only first-order accurate. Second-order spatial accuracy is achieved by using an extension of van Leer's [25] MUSCL technique to unstructured meshes; this approach consists in applying a piecewise linear interpolation to obtain the states W_{ij} and W_{ji} at the interface between control volumes C_i and C_j:

$$\tilde{W}_{ij} = \tilde{W}_i + \frac{1}{2}\left(\overrightarrow{\nabla \tilde{W}}\right)_i . \overrightarrow{s_i s_j}, \quad \tilde{W}_{ji} = \tilde{W}_j - \frac{1}{2}\left(\overrightarrow{\nabla \tilde{W}}\right)_j \cdot \overrightarrow{s_i s_j}, \tag{5}$$

where $\tilde{W} = (\rho, u, v, p', k, \varepsilon)^T$ – in other words, the interpolation is performed on the physical variables instead of the conservative variables. The approximate nodal gradient $(\nabla \tilde{W})_i$ is obtained by averaging the Galerkin gradients computed on each triangle of C_i:

$$\left(\overrightarrow{\nabla \tilde{W}}\right)_i = \frac{\int\int_{C_i} \left(\overrightarrow{\nabla \tilde{W}\,|_T}\right) \mathrm{d}\vec{x}}{\int\int_{C_i} \mathrm{d}\vec{x}} = \frac{1}{\mathrm{area}(C_i)} \sum_{T \in C_i} \frac{\mathrm{area}(T)}{3} \sum_{j \in T} \tilde{W}_j \left(\overrightarrow{\nabla \phi_j^T}\right). \tag{6}$$

The construction given by Eqs. (5) and (6) results in a half-upwind (Fromm-like) scheme which is spatially second-order accurate but may present spurious oscillations in the solutions, expressing a loss of monotony. One way to circumvent this problem is to make a compromise between the first-order scheme and the second order one by applying slope limitation procedures. We refer to Fezoui and Dervieux [4] for a detailed description of the application of several limitation procedures in the context of the present finite element/finite volume MUSCL method.

2.3. Time integration

It is well known that a global Newton iteration starting from arbitrary initializations cannot be applied to compressible flows. In order to approach the convergence domain of a (modified) Newton iteration, an Euler implicit time advancing is constructed with the following features (see [5]).

- *Linearization*: The linearization is obtained by freezing the Jacobian in Roe's flux difference splitting for the convective fluxes, and by exact differentiation for the laminar and turbulent diffusive fluxes. For the source term, only the dissipative part of the source term is linearized, and this is obtained by exact differentiation. In addition, the turbulent viscosity μ_t is frozen so that the k and ε variables are coupled to each other but uncoupled from the mean flow variables. Concerning the wall functions, both slip and shear stress conditions are linearized [8].
- *Preconditioning*: It is performed by using first-order Godunov scheme because it is tridiagonal in 1D and better conditioned than the second-order one, which furthermore produces larger matrix bandwidths (pentadiagonal in 1D).
- *Local time stepping and CFL law*: A local time step is computed on each cell from the Courant number (denoted by "CFL" in the sequel). The CFL number can be for instance an increasing function of time taken as the inverse of the non-linear residual (L_2 norm), in order to ensure the progressive switch from the unsteady phase to the asymptotic convergence.

- Each time step involves the solution of the two linear systems (mean flow and turbulent variables). Approximate solutions to these systems are obtained by using a linear multigrid strategy by volume agglomeration which is described in the next section.
- Each calculation is started from a uniform flow and driven to at least a 10^{-5} residual reduction.

3. The linear multigrid algorithm

The multigrid method adopted here is based on the use of macroelements (macrocontrol volumes) which form the coarse discretizations of the computational domain. It is an extension of the linear multigrid approach developed by Mulder [20] and Lallemand et al. [10] to accelerate the solution of linear systems.

3.1. Grid coarsening by agglomeration

In [1,2] 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 macroelements of the coarser level, according to the following steps:

```
FOR each control volume $C_i$ in $\Omega$ DO
   IF $C_i$ has already been included in a group the
      Consider the next control volume
   ELSE
      Create a new group containing $C_i$
      Put into this group the neighboring control volumes of $C_i$
      which do not already belong to another existing group
      IF the new created group contains only $C_i$ THEN
         Destroy this group and put $C_i$ in an existing group
         containing at least one of the neighbors of $C_i$
      ENDIF
   ENDIF
   GOTO the next control volume
ENDFOR
```

The main advantage of this method is that it allows an automatic generation of the coarser discretizations without building any coarse triangulation. However, this strategy is also strongly dependent on the numerotation of mesh vertices in the original fine mesh (remember that fine mesh vertices are also fine mesh control volumes). Finally, the present strategy is isotropic in nature and therefore behaves poorly in the presence of highly stretched meshes.

3.2. Coarse grid approximation for convective terms

We 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 macroelements. 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. Both conservative variables and normal vectors are interpolated between the different grids. The coarse grid variables are deduced by transfer operators (defined later). The normal vectors, linked with each coarse macrocontrol volume, result from the summation of the finer grid vectors (for the fine mesh control volumes that have a common boundary with the macroelements); as a result, at most one flux is computed between two macrocontrol volumes.

3.3. Coarse grid approximation for diffusive terms

To evaluate the diffusive laminar and turbulent terms on a coarse level, related basis functions are needed. We recall here the approach proposed by Koobus et al. [9] (see also Carré [1] for more details). In the finite element formulation on the fine grid, the equations are integrated and assembled by edges (convective terms) and triangles (diffusive terms and nodal gradients). As triangles do not exist on the coarser grids, it is necessary to define a new formulation for the calculation of diffusive terms. A Galerkin approximation for any function f_h can be expressed by

$$f_h(x,y) = \sum_i f_i \phi_i(x,y),$$

where f_i is the value of f_h at mesh vertex $s_i = (x_i, y_i)$ and ϕ_i is the nodal basis function associated to s_i and assembled from the P1 basis functions defined on each triangle (element) that takes part in the finite element support of s_i. The finite element formulation applied to the diffusive laminar and turbulent terms shows that the general form of the resulting discrete quantities depends on the integrals:

$$a_{lm}^{ij} = \int\int_\Omega \frac{\partial \phi_i}{\partial x_l} \frac{\partial \phi_j}{\partial x_m} \, \mathrm{d}\vec{x},$$

where $l, m \in \{1,2\}$, $x_1 = x$ and $x_2 = y$. The integrals a_{lm}^{ij} are calculated on the fine grid and assembled by points (case $i = j$) and edges (case $i \neq j$). A summation of these quantities by neighboring relations on virtual coarse points, or edges connecting two macroelements, allows to define the coarse grid integrals:

$$a_{lm}^{IJ} = \int\int_\Omega \frac{\partial \phi_I}{\partial x_l} \frac{\partial \phi_J}{\partial x_m} \, \mathrm{d}\vec{x},$$

where

$$\int\int_\Omega \frac{\partial \phi_I}{\partial x_l} \frac{\partial \phi_J}{\partial x_m} \, \mathrm{d}\vec{x} = \sum_{i \in I, j \in J} \int\int_\Omega \frac{\partial \phi_i}{\partial x_l} \frac{\partial \phi_j}{\partial x_m} \, \mathrm{d}\vec{x}.$$

Unfortunately, the above coarse grid formulation is not consistent with the standard finite element formulation of the fine grid approximation. In fact, the resulting coarse basis functions resulting from a summation of fine grid ones are depicted in Fig. 3 in a 1D point of view, and it is clear that the coarse grid formulation is not able to approximate accurately the fine grid problem. Consequently the above formulation is completed by a correction factor first proposed in [9] for approximately preserving the consistency between two grids. This factor, multiplying each diffusive flux component, remains constant for each application and is expressed from

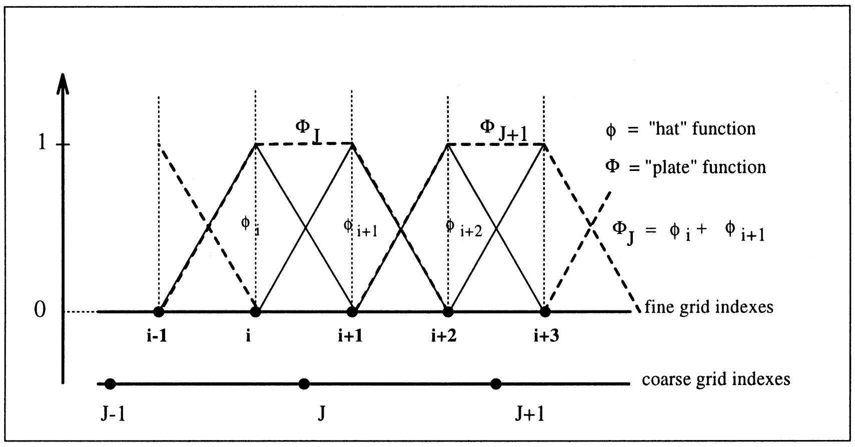

Fig. 3. Comparison of 1D basis functions of the fine and coarse grids.

$$K_N = \frac{2(N-1)^2}{(2N-1)^2} \quad \text{with} \quad N = \sqrt[d]{N_z}, \tag{7}$$

where d is the space dimension and N_z the total number of coarse macroelements.

3.4. Multigrid solution scheme

In the linear multigrid method (or correction scheme) the corrections of the fine grid solution are computed on the coarse grids. On the finest grid, we solve partially a linear system resulting from the linearization of the equations given in Section 2. Let us denote

$$\left(\frac{\text{area}(C_i)}{\Delta_t} Id + \Phi' + \Psi' + \tilde{\Psi}' + \Omega'\right)\delta W^{n+1} = \Phi + \Psi + \tilde{\Psi} + \Omega, \tag{8}$$

where Δ_t is the time increment, Φ, Ψ, $\tilde{\Psi}$ and Ω, respectively, represent the convective, the diffusive (laminar and turbulent) and the source term explicit fluxes. On the other hand, Φ', Ψ', $\tilde{\Psi}'$ and Ω' are the corresponding Jacobian matrices (calculated from the first order scheme for the convective term). We can rewrite the above linear system as follows:

$$M(W^n)\delta W^{n+1} = b, \qquad \delta W^{n+1} = W^{n+1} - W^n, \tag{9}$$

where δW^{n+1} corresponds to the approximated solution. Futhermore, we define the residual R which gives the right-hand side of the linear system to be solved on the coarse grid:

$$R = b - M(W^n)\delta W^{n+1}. \tag{10}$$

The error term, obtained by the solution of the residual equation (coarse grid system), is prolongated to correct the solution on the fine grid. The algorithm consists in solving $MX = b$ and can be written as (multigrid algorithm for one V-cycle with Jacobi smoother):

- X^α: αth multigrid solution approaching δW^{n+1} on the fine grid with $X^0 = 0$,
- $X^0_{(k)}$: initial solution of the relaxation method for solving the system on G_k, for $k = 1, \ldots, N$ (where N is the number of levels),
- $p_{k+1,k}$ and $r_{k,k+1}$, respectively, denote the prolongation and restriction operators (defined later),
- ν_1 and ν_2 are the numbers of pre- and post-smoothing,
- ω is the relaxation factor of the iterative method,
- $M_{ij}(k) = I + m_{ij}(k)$ is the matrix obtained by the discretization on G_k,
- $b_i(k)$ is the right-hand side of system on G_k,
- $I = \frac{\text{area}(C_i)}{\Delta_t} Id$.

Step 1 (*Initialization*): $X^0_{(1)} = X^\alpha$ (solution obtained at the previous V-cycle).
Step 2 (*Pre-smoothing and calculation of corrections*): For $k = 1, \ldots, N$.

$$X^0_{(k)} = \begin{cases} 0 & \text{if } k > 1, \\ X^\alpha & \text{if } k = 1. \end{cases} \tag{11}$$

The following nodewise 4×4 block Jacobi iteration is applied:

$$\begin{aligned} \overline{X^l_i(k)} &= (I + m_{ii})^{-1}_{(k)} \left[b_i(k) - \sum_{j \neq i} m_{ij(k)} X^{l-1}_{j(k)} \right], \quad l = 1, \ldots, \nu_1, \\ X^l_{i(k)} &= (1-\omega) X^{l-1}_{i(k)} + \omega \overline{X^l_i(k)}, \\ b_{i(k+1)} &= r_{k,k+1}\left(b_i(k) - \sum_{j \neq i, j = i} m_{ij(k)} X^{\nu_1}_{j(k)} \right), \end{aligned} \tag{12}$$

where $b_{(k+1)}$ is the restricted residual from level k to level $k+1$ and $m_{ij}(k)$ represents a discretization of the continuous operator on the level k.

Step 3 (*Post-smoothing and prolongation of corrections*): For $k = N-1,\dots,1$

$$
\begin{aligned}
&X_{i(k)} = X_{i(k)}^{\nu_1} + p_{k+1,k} X_{i(k+1)},\\
&X^0_{i(k)} = X_{i(k)},\\
&\overline{X^l_{i(k)}} = (I + m_{ii})^{-1}_{(k)} \left[b_{i(k)} - \sum_{j \neq i} m_{ij(k)} X^{l-1}_{j(k)} \right], \quad l = 1,\dots,\nu_2,\\
&X^l_{i(k)} = (1-\omega) X^{l-1}_{i(k)} + \omega \overline{X^l_{i(k)}},\\
&X_{i(k)} = X^{\nu_2}_{i(k)},
\end{aligned}
\tag{13}
$$

where $p_{k+1,k}X_{k+1}$ represents the projected correction from level $k+1$ to level k (coarse grid correction).

Step 4 (*Final update*): $X_i^{\alpha+1} = X_i(1)$.

3.5. Transfer operators

A condition to obtain multigrid efficiency [6] is that the summation of the orders of the transfer operators is greater than the order of the partial differential equation to be solved. This condition, developed in [26,7], 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 an agglomeration context. In our case, we keep the same order for both restriction and prolongation which is in accordance with the previous condition only for the convective approximation, but allows building simple and diagonally dominant coarse grid matrices.

We define $[W_k]$ as the solution on the kth level, we note the value $[W_k](Z_l^k)$ in the (Z_l^k) system of level k $(l = 1,\dots,N_k)$.

- *Restriction operator* for $[W_k]$ solutions: $r^s_{k,k+1}$ $(j = 1,\dots,N_{k+1})$

  $$\left[r^s_{k,k+1}(W_k)\right]\left(Z_j^{k+1}\right) = \frac{\sum_{l \in U^k(j)} \operatorname{area}\left(Z_l^k\right) W_k\left(Z_l^k\right)}{\operatorname{area}\left(Z_j^{k+1}\right)}, \tag{14}$$

  where $U^k(j)$ denotes the set of the l indexes of Z_l^k zones on the level k with $Z_l^{k+1} = \bigcup_{l \in U^k(j)} Z_l^k$. This operator transfers the solution from level k to level $k+1$ in order to build only the discretization of the continuous operator on this level and is not used in the multigrid cycle. It is a weighted approximation of the level k solution.
- *Restriction operator* for the right-hand side: $r^{rhs}_{k,k+1}$. The RHS on level $k+1$ is obtained by summation of RHS from level k. It is defined as

  $$\left[r^{rhs}_{k,k+1}(b_k)\right]\left(Z_j^{k+1}\right) = \sum_{l \in U^k(j)} b_k\left(Z_l^k\right). \tag{15}$$

- *Prolongation operator* for the error term: $p^s_{k+1,k}$. This operator transfers the correction e_{k+1} from level $k+1$ to level k. It is defined by a trivial injection between the two grids:

  $$\left[p^s_{k+1,k}(e_{k+1})\right]\left(Z_j^k\right) = e_{k+1}\left(Z_l^{k+1}\right), \qquad l = 1,\dots,N_k. \tag{16}$$

- *Averaging operator*: Q_k. In [9], it has been found useful to apply an averaging operator to the computed error term. This computation allows to correctly smooth the total error components before updating the solution. The value of the weighting coefficient β of the prolongation operator is set to 1, and has been chosen after several experiments. In this case, the error component is averaged with its neighboring values and this operation allows to increase the robustness of the present multigrid method.

  $$Q_k(e_k) = \beta \operatorname{averg}(e_k) + (1-\beta) e_k, \tag{17}$$

where

$$[\mathrm{averg}(e_k)]_i = \frac{1}{\mathrm{areat}} \sum_{j \in N^k(i) \cup i} \mathrm{area}(j) e_{k,j}, \quad \mathrm{areat} = \sum_{j \in N^k(i) \cup i} \mathrm{area}(j) e_{k,j}.$$

Finally

$$\overline{p}^s_{k+1,k} e_{k+1} = Q_k . p^s_{k+1,k}(e_{k+1}).$$

4. Parallelization strategy

4.1. Basic parallelization strategy

The parallelization strategy adopted for the 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 [3] as well as in 3D [11]. The underlying mesh is assumed to be partitioned into several submeshes, each defining a subdomain. Basically the same "old" serial code is going to be executed within every subdomain. Applying this parallelization strategy to the previously described flow solver results in modifications occuring in the main time-stepping loop in order to take into account one or several assembly phases of the subdomain results, depending on the order of the spatial approximation and on the nature of the time advancing procedure (explicit/implicit). The assembly of the subdomain results can be implemented in 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 to the mixed finite volume/element formulation considered herein. 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 [11]. In the present study we will consider one triangle wide overlapping mesh partitions for second order accurate implicit computations.

4.2. Parallel coarsening strategy

The parallelization of the linear multigrid solution procedure described in Section 3 requires a preliminary step aiming at the construction of appropriate data structures for the distribution of coarse grid calculations. Here, this has been achieved by developing a parallel variant of the original "greedy" type coarsening algorithm (see Section 3.1) which now includes additional communication steps for a coherent construction of the communication data structures on the partitioned coarse grids. The choice of developping a fully parallel coarsening algorithm rather than building a sequential (and probably uncoupled) preprocessing tool, was mainly motivated by a potential future utilization of the resulting solution strategy in the context of deforming or adaptive meshes.

As stated earlier, the intra-level parallelization is based on the use of overlapping mesh partitions. The overlapping layer is one triangle wide. In order to build the additional communication data structures that are required for the parallelization of coarse grid calculations, we define a notion of submesh property. We consider that a submesh is the owner of all the mesh vertices it contains except those that are placed on the exterior side of artificial boundaries. The latter are indeed owned by the neighboring submeshes. By "owner" we mean that the submesh is responsible for the final update of the physical value attached to the corresponding vertices. In the parallel agglomeration strategy, we keep these notions of submesh property and submesh neighborhood; the main steps are the following:

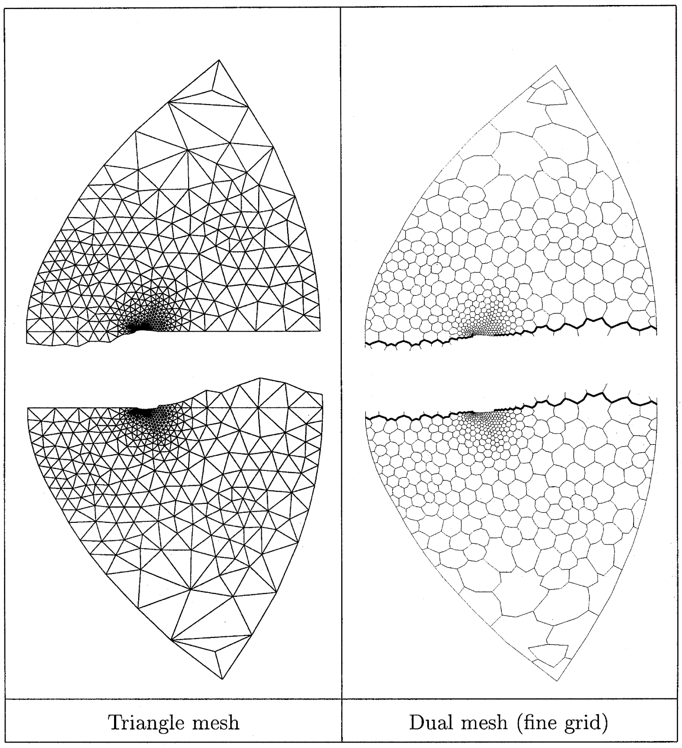

Fig. 4. Parallel coarsening strategy: first level.

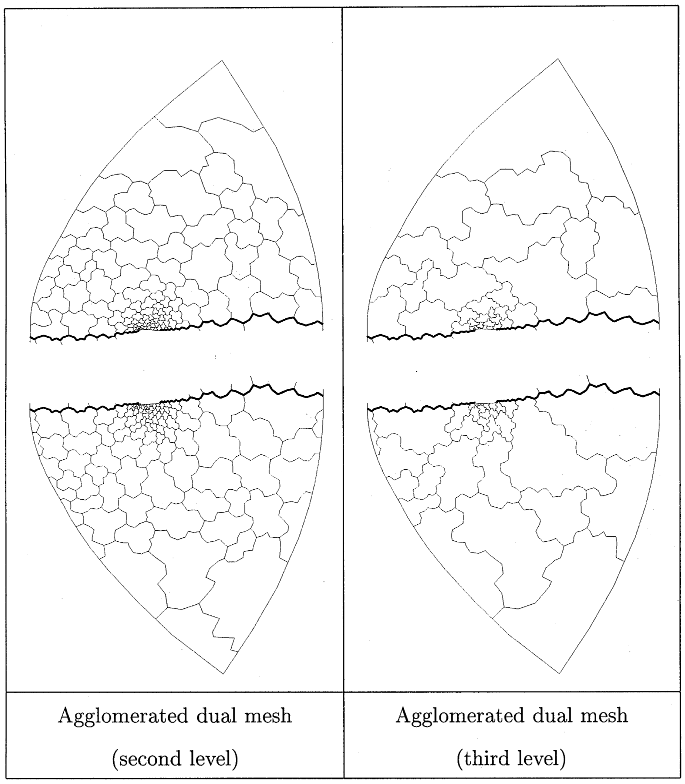

Fig. 5. Parallel coarsening strategy: second and third levels.

```
FOR each submesh DO IN PARALLEL
    Perform standard sequential agglomeration on the owned vertices
    Send to neighbors the result of the agglomeration
    Receive from neighbors the result of their agglomerations
    Reproduce the neighboring agglomerations on the non-owned vertices
    Construct appropriate communication data structures
ENDFOR
```

In Figs. 4 and 5, we can observe the conservation of the fine mesh partitioning and the coherence of the overlapping layer on the agglomerated submeshes.

5. Numerical and performance results

5.1. Parallel performances for steady flow calculations

Performance results are given for 64 bit arithmetic computations. In Tables 1–11, N_g is the number of grids (fine mesh included), N_c denotes the number of V-cycles used for each linear system solution and N_p is

Table 1
Characteristics of the NACA0012 meshes

MESH	# Vertices	# Triangles	# Edges
M1	48 792	96 896	145 688
M2	194 480	387 584	582 064

Table 2
Parallel performance results on a FastEthernet Pentium Pro cluster Calculations with mesh M1: $\nu_1 = \nu_2 = 2$

N_g	N_c	N_p	Elapsed (s)	CPU (s)	% CPU	$S(N_p)$
4	4	2	2160	2107	97	1.00
4	4	4	1229	1167	95	1.75
4	4	6	908	844	93	2.35
4	4	8	705	628	89	3.05
5	2	2	1611	1570	97	1.00
5	2	4	927	875	94	1.75
5	2	6	688	632	92	2.35
5	2	8	540	475	88	3.00

Table 3
Parallel performance results on a SGI Power Challenge Array Calculations with mesh M1: $\nu_1 = \nu_2 = 2$

N_g	N_c	N_p	Elapsed (s)	CPU (s)	% CPU	$S(N_p)$
4	4	1	2089	2075	99	1.00
4	4	2	1072	1065	99	1.95
4	4	4	554	550	99	3.80
4	4	6	426	422	99	4.90
4	4	8	383	381	99	5.45
5	2	1	1462	1455	99	1.00
5	2	2	760	756	99	1.95
5	2	4	392	389	99	3.75
5	2	6	303	301	99	4.85
5	2	8	261	258	99	5.60

Table 4
Communication costs on a `FastEthernet Pentium Pro` cluster: $N_p = 8$ calculations with mesh M1: $\nu_1 = \nu_2 = 2$

N_g	N_c	N_{iter}	Elapsed (s)	CPU (s)	% CPU	Comm	% Comm
4	2	128	736	656	89	68	9
5	2	89	540	475	89	63	12
6	2	89	550	492	89	73	13
7	2	89	567	507	89	87	15

Table 5
Communication costs on a `SGI Power Challenge Array`: $N_p = 8$ calculations with mesh M1: $\nu_1 = \nu_2 = 2$

N_g	N_c	N_{iter}	Elapsed (s)	CPU (s)	% CPU	Comm	% Comm
4	2	128	387	385	99	30	8
5	2	89	272	270	99	25	9
6	2	89	280	278	99	29	10
7	2	89	283	280	99	33	12

Table 6
Parallel performance results on a `FastEthernet Pentium Pro` cluster calculations with mesh M2: $\nu_1 = \nu_2 = 4$

N_g	N_c	N_p	Elapsed (s)	CPU (s)	% CPU
6	1	12	1944	1693	87

Table 7
Parallel performance results on a `SGI Power Challenge Array` calculations with mesh M2: $\nu_1 = \nu_2 = 4$

N_g	N_c	N_p	Elapsed (s)	CPU (s)	% CPU	$S(N_p)$
6	1	2	4184	4166	99	1.00
6	1	4	2186	2173	99	1.90
6	1	6	1650	1643	99	2.55
6	1	8	1384	1377	99	3.00

Table 8
Simulations on a `FastEthernet Pentium Pro` cluster: $N_p = 12$ comparison of multigrid and single grid algorithms using mesh M2

N_g	N_c	$\nu_f - \nu_1/\nu_2$	ε	N_{iter}	Elapsed (s)	CPU (s)	% CPU
1	–	∞	10^{-1}	125	9 h 28 mn	8 h 24 mn	88
1	–	∞	10^{-2}	117	9 h 40 mn	8 h 48 mn	91
1	–	350	10^{-10}	178	9 h 10 mn	8 h 17 mn	90
1	–	400	10^{-10}	157	9 h 06 mn	8 h 14 mn	90
1	–	450	10^{-10}	142	9 h 28 mn	8 h 20 mn	88
6	∞	4/4	10^{-1}	117	57 mn	50 mn	88
6	∞	4/4	10^{-2}	116	1 h 56 mn	1 h 42 mn	88
6	1	4/4	10^{-10}	117	33 mn	29 mn	87

Table 9
Characteristics of the RAE2822 meshes

MESH	# vertices	# triangles	# edges
MT1	8220	16000	24220
MT2	32440	64000	96440

Table 10
Parallel performance results on a SGI Power Challenge Array turbulent flow around the RAE2822 airfoil

MESH	N_g	N_c	N_p	Elapsed (s)	CPU (s)	% CPU	$S(N_p)$
MT1	4	2	1	499	496	99	1.00
–	4	2	2	292	290	99	1.70
–	4	2	4	188	186	99	2.65
–	4	2	6	168	168	99	3.00
MT2	5	2	1	3352	3333	99	1.00
–	5	2	2	1811	1810	99	1.85
–	5	2	8	685	680	99	5.90

Table 11
Parallel performance results on a SGI Origin 2000 Euler flow around an ONERA M6 wing

N_p	Elapsed C3	CPU C3	$S(N_p)$ C3	Elapsed MG	CPU MG	$S(N_p)$ MG	$G(N_p)$
2	10260	10210	1.00	3092	3076	1.00	3.3
4	4555	4530	2.25	1586	1578	1.95	3.0
8	2078	2066	5.00	680	675	4.55	3.0

the number of processes for the parallel execution; “Elapsed” denotes the total elapsed execution time and “CPU” denotes the total CPU time (taken as the maximum value over the local measures); “Comm” denotes the communication time (local exchange steps at artificial boundaries and global reduction operations); finally, the speedup $S(N_p)$ is calculated using the elapsed execution times. Calculations have been performed on a SGI Power Challenge array equipped with Mips R10000/180 MHz processors and on a cluster of Pentium Pro P6/200 MHz running the Linux operating system. In the latter case, the same code has been compiled using the G77 gnu compiler with maximal optimization options. The cluster nodes are currently interconnected via a 100 Mbit/s FastEthernet switch. Communications are handled using MPICH (version 1.1.0).

5.1.1. *Laminar flow around a* NACA0012 *airfoil*

The test case under consideration is given by the external flow around a NACA0012 airfoil at a free-stream Mach number of 0.8, a Reynolds number equal to 73 and an angle of incidence of 10°. Two embedded triangulations have been constructed; their characteristics are summarized in Table 1. The solution is visualized in Fig. 6 in terms of the steady Mach lines obtained with mesh M1.

Parallel efficiencies: Tables 2 and 3 compare performance timings for calculations performed using mesh M1. In these tables ν_1 and ν_2, respectively, denote the number of pre- and post-smoothing steps (Jacobi relaxations); the V-cycle has been selected here. For $N_g = 4$ and $N_c = 4$ the number of non-linear iterations to steady state is equal to 89 (normalized residual reduced to 10^{-10}); for the case $N_g = 5$ and $N_c = 2$ this figure is equal to 88. In Table 2 the parallel speed-up is computed relatively to the $N_p = 2$ measure. Tables 4

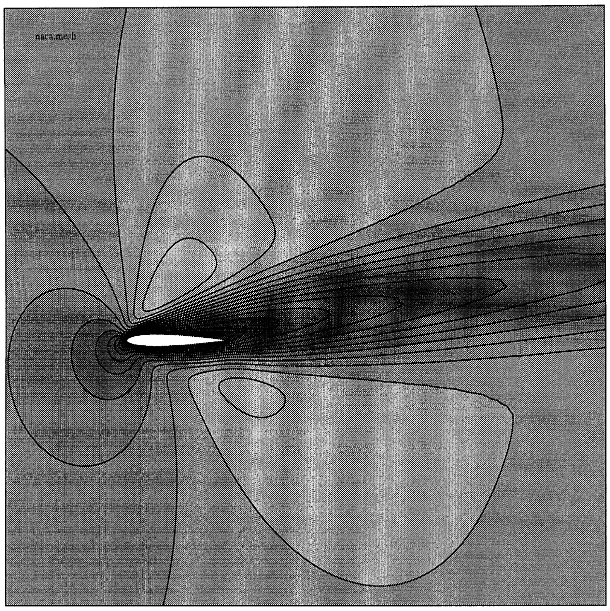

Fig. 6. Steady Mach lines for the viscous flow around the NACA0012 airfoil.

and 5 give communication timings for a fixed number of multigrid cycles and various numbers of levels using mesh M1; in these tables, N_{iter} denotes the number of non-linear iterations to steady state. Tables 6 and 7 compare performance timings for calculations performed using mesh M2 and with $N_g = 6$ and $N_c = 1$; the number of non-linear iterations to steady state is equal to 117 (normalized residual reduced to 10^{-10}).

The first comment we shall make concerning these results is that the cluster computing approach is a viable alternative to true parallel computing on an integrated MIMD system. Excellent parallel efficiencies are obtained with mesh M1 for a number of machines up to 6. We also note that for mesh M2, the CPU utilization is as high as 87% on 12 machines; unfortunately we have not been able to use less that 12 machines on this case due to memory requirements.

On the other hand, the performance results obtained on the `SGI Power Challenge Array` are not so satisfying. The parallel speed-up using 8 processors evolves from 5.6 (mesh M1) to 6.0 (mesh M2) which is a lower improvement than we could expect by increasing the global problem size by a factor of about 4. This behaviour can be partially explained by referring to the hardware architecture of the `SGI Power Challenge Array`, and more particularly to the interconnection system. The parallel executions considered here take place within one `POWERnode`. Therefore, inter-processor communication steps consist in shared memory read/write operations via the so-called `POWERpath-2` transaction bus. The maximal bandwith sustainable by this system is 1.2 Gb/s. However, when using mesh M2 and a number of grids $N_g = 6$, the overall memory requirement is actually exceeding 1.2 Gb which is probably synonym of memory bandwith saturation. In order to illustrate further our point of view, we report on a set of complementary results of simulations that have been performed on both the `SGI Power Challenge Array` and the `SGI Origin 2000` systems. In our case, the `SGI Origin 2000` system is based on `Mips R10000/195 MHz` processors with 4 Mb of cache memory (instead of 2 Mb on the `Mips R10000/180 MHz` used in the `SGI Power Challenge Array`). Mesh M2 has been used: for $N_g = 1$ (i.e. single grid algorithm) we perform 12 Jacobi relaxations for each linear system solution; for $N_g \in [2, 6]$ we perform one V-cycle with 4 pre- and post-smoothing steps (i.e. $\nu_1 = \nu_2 = 4$). Calculations have been done on for 4 and 8 processors for the two parallel systems and timings have been measured for 20 time steps. Fig. 7 depicts the efficiency curve (the efficiency is computed as $S(N_p)/p$ where the parallel speed-up is relative to the elapsed time on 4 processors). On the `SGI Power Challenge Array` the efficiency is rather low but stable accross the number of levels in the multigrid hierarchy (between 79% and 81%). On the `SGI Origin 2000` a superlinear speed-up is observed for each value of N_g and the efficiency is decreasing (from 127% for $N_g = 1$

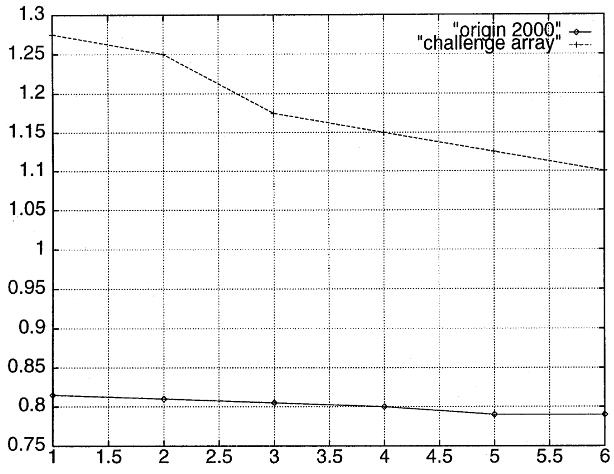

Fig. 7. Parallel efficiencies between 4 and 8 processors.

to 110% for $N_g = 6$) which is more in accordance with the results of Table 5 that were obtained using the coarser mesh M1.

Multigrid versus single grid: We are interested here in comparing the single grid and the multigrid approaches when solving the steady laminar viscous flow under consideration using mesh M2. Concerning the single grid algorithm, the objective is to choose appropriate values for the number of relaxation steps and the tolerance on the linear residual so that a good compromise is obtained between the number of non-linear iterations (pseudotime steps) to convergence and the corresponding elapsed time. For both algorithms, the time step is calculated using to the law $\mathrm{CFL} = \min(500 \times it, 10^6)$ where *it* denotes the non-linear iteration. Table 8 compares results of various simulations performed on a 12 nodes Pentium Pro cluster. In this table, ∞ means that the number of fine mesh Jacobi relaxations (ν_f) or the number of multigrid *V*-cycles (N_c) has been set to an arbitrary large value such that the linear resolution is driven until the prescribed residual reduction (ε) is obtained; ν_1 and ν_2 denote the number of pre- and post-smothing steps (Jacobi relaxations) when using the multigrid algorithm. We observe that the non-linear convergence of the single grid is optimal when driving the linear resolution to a two decade reduction of the normalized linear residual. However the

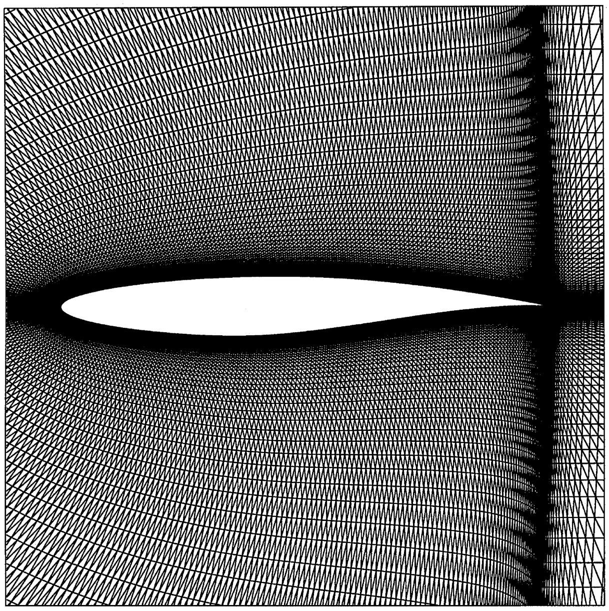

Fig. 8. Zoom on the mesh near the RAE2822 airfoil.

corresponding elpased time is minimized when fixing the number of fine mesh relaxations to 400. On the other hand, one V-cycle with 4 pre- and post-smoothing steps is sufficient for an optimal convergence of the multigrid algorithm. Comparing the two entries of Table 8 corresponding to the case $\varepsilon = 10^{-1}$, it is seen that the multigrid algorithm yields a non-linear convergence in 117 time steps instead of 125 time steps for the single grid algorithm. This clearly shows that for the same level of linear convergence, the non-linear solutions obtained with the two algorithms at a particular time step, are not strictly equivalent; in particular, the increment δW^{n+1} (see Eq. (9)) resulting from the multigrid algorithm is bringing improvements on both the high and low frequency components of the error which in turns contribute to a higher quality physical solution. We conclude this section by noting that the multigrid algorithm is about 16 times faster than the single grid algorithm on the present test case and considering a large probleme size (about 0.76 million unknown).

5.1.2. *Turbulent flow around a* RAE2822 *airfoil*

The next test case consists in the turbulent flow over a RAE2822. This flow is defined by a freestream Mach number of 0.73, a Reynolds number of 6.5×10^6, and an incidence angle of 2.79°. Two embedded C type triangulations have been constructed; their characteristics are summarized in Table 9. A zoom of mesh

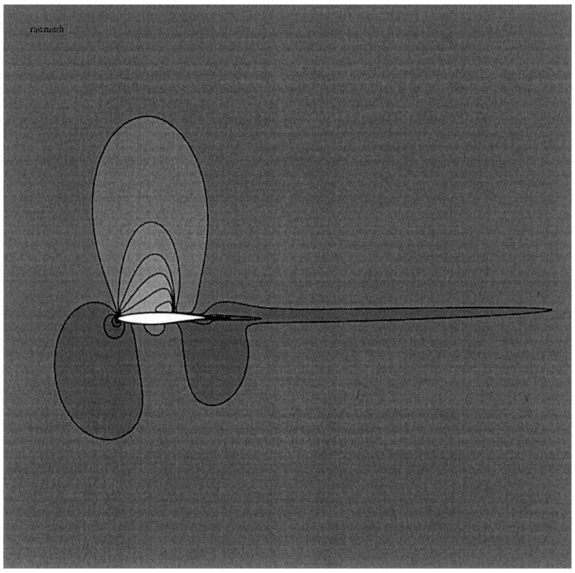

Fig. 9. Steady Mach lines for the turbulent flow around the RAE2822 airfoil.

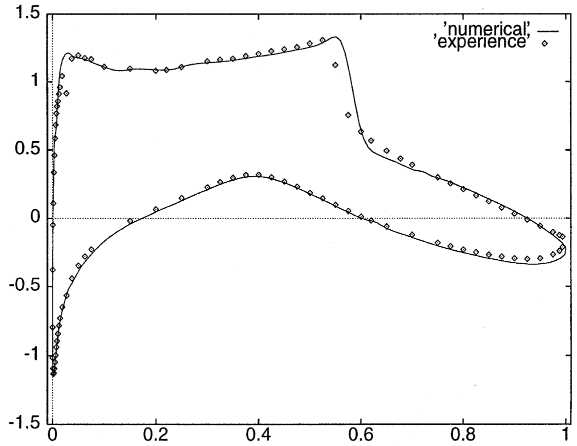

Fig. 10. Comparison of computed surface pressure with experimental measurements.

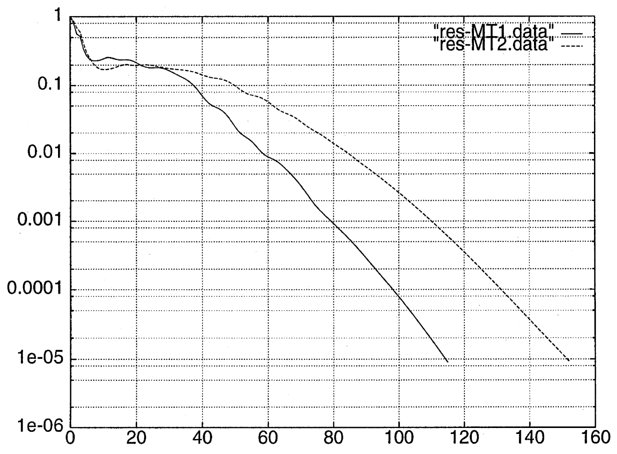

Fig. 11. Non-linear convergence for the turbulent around the RAE2822 airfoil.

MT2 near the airfoil is shown in Fig. 8. The solution is visualized in Fig. 9 in terms of the steady Mach lines obtained with mesh MT2. Fig. 10 compares computed surface pressure with available experimental measurements. The steady flow is obtained using the multigrid algorithm based on the *F*-cycle with $N_g = 4$ and $\nu_1 = \nu_2 = 3$ for mesh MT1, and $N_g = 5$ and $\nu_1 = \nu_2 = 4$ for mesh MT2. In each case the number of cycles N_c is fixed to 2. The time step is calculated using to the law $\mathrm{CFL} = \min(2 \times it, 800)$ where *it* denotes the non-linear iteration. The non-linear convergence is shown on Fig. 11 in terms of the total energy normalized residual. Performance results are given in Table 10.

6. Multigrid by agglomeration for industrial applications

6.1. *The* N3S-NATUR *solver*

The work described here is concerned with the implementation and evaluation of a parallel linear multigrid strategy in the N3S-NATUR industrial CFD package. The N3S-NATUR software package is the object of an ongoing coordinated effort to develop a parallel solver dedicated to the numerical simulation of industrial 3D compressible steady or transient flows. This activity is supported by a consortium consisting of three indutrial partners which are end-users and co-developers of N3S-NATUR (EDF, SNECMA and RENAULT), two software companies (SIMULOG and METRAFLU) and two research institutions (ECOLE CENTRALE DE LYON and INRIA). The version V1.2 of N3S-NATUR is characterized by the following features (see [2] for more details):

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

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

(c) *Numerical features*: The Navier–Stokes equations are solved in conservative form. The discretization in space relies on a mixed finite volume (for the convective terms)/finite element (for the diffusive terms) formulation as the one described in Section 2. The convective terms are computed using an approximate Riemann solver (Roe's or van Leer's method for perfect gases, modified Roe's solver for real gases, preconditioned Roe–Turkel's solver for low Mach number flows) and the MUSCL (Monotonic

Upstream Scheme for Conservation Law) technique is used for the extension to second order accuracy. Unsteady flows based on deforming meshes can be handled thanks to an appropriate calculation of the convective terms. Explicit or linearized implicit time integration techniques are available with (for each case) the possibility to choose between first or second order in time accuracy. Jacobi and Gauss–Seidel relaxations are implemented for the solution of the linear systems resulting from the implicit time integration scheme.

The parallelization of N3S-NATUR is described in details in [12]. The adopted strategy combines mesh partitioning techniques and a message-passing programming model. The MS3D [15] tool is used to partition the mesh and generate the appropriate input files (local data structures for submesh topology and data exchange at submesh interfaces). The parallelization of the multigrid part is based on the use of one tetrahedra wide partitions and extends the work described in Section 4. The communication steps are implemented using MPI.

There are two practical situations in the N3S-NATUR solver that call for the solution of large sparse linear systems: the linearized implicit time integration scheme (a maximum of three systems are obtained in this case, one for the mean flow variables, one for the turbulent variables and one for the chemical species) and the mesh deformation procedure (a pseudostructural method is used to determine the new coordinate positions for internal vertices given a prescribed position of the boundary vertices). In this study, the linear multigrid strategy by volume agglomeration described in Section 3 is introduced in N3S-NATUR in order to allow for an efficient treatment of all these systems.

6.2. Applications

The implementation of the linear multigrid algorithm in N3S-NATUR has been done and validated for steady laminar flows; ongoing efforts concern the extension of this methodology to steady and unsteady turbulent flows. We present here preliminary results of the application of this methodology to the classical test case of the Euler flow around an ONERA M6 wing at a free stream Mach number of 0.84 and an angle of attack equal to 3.06°. The underlying tetrahedral mesh contains 115 351 vertices, 643 392 tetrahedra and 774 774 edges. The CFL number has been fixed to 10^9. The calculation is started from a uniform flow. The single grid implicit algorithm has been tested for the following situations: 60 (case C1), 70 (case C2) and 75 (case C3) Jacobi relaxations are performed at each time step. For the multigrid algorithm, 3 coarse levels have been constructed (i.e. $N_g = 4$) and a constant number of $N_c = 2$ V-cycles, with $\nu_1 = 2$ pre-smoothing and $\nu_2 = 3$ post-smoothing steps, has been used at each time step. The non-linear convergence curves corresponding to these four situations are visualized in Fig. 12. These curves show that the single grid case C3 leads to a non-linear convergence similar to the one obtained when using the multigrid algorithm. We will therefore retain the case C3 for further comparisons of the two algorithms.

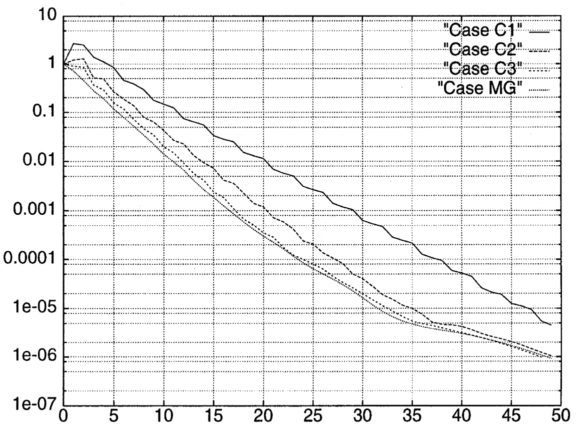

Fig. 12. Non-linear convergence for the Euler flow around an ONERA M6 wing.

Calculations have been peformed on a `SGI Origin 2000` system. Performance results are given in Table 11 where the parallel speed-up $S(N_p)$ is calculated using the total elapsed time, relatively to the case $N_p = 2$; $G(N_p)$ denotes the overall gain between the multigrid and the single grid algorithms. For $N_p = 8$, we observe a 10% degradation of the parallel speed-up (and of the overall gain between the two algorithms) which is attributed to the redundant arithmetic operations incurred by the partitioning strategy selected here. This behavior is consistent with what was experimented using a simplified kernel extracted from `N3S-MUSCL`, the previous generation of the `N3S-NATUR` software (see [11] for more details). We finally note that the gain between the two algorithms is well below what was obtained in the 2D case (3 instead of 16). This figure is nevertheless satisfying knowing that the flow under consideration is relatively simple and that the equivalent 2D size of the underlying mesh is $115\,351^{2/3} \approx 2370$ vertices.

7. Conclusion and future works

Parallel computing offers the opportunity to simulate compressible flows of increasing physical complexity, around or withing complex geometries. However, designing robust and efficient solvers on unstructured finite element meshes is still a challenging objective that should not be neglected to the profit of simple solution methods exhibiting high level of parallel efficiency.

The objective of the work presented in this paper is to develop a flow solution technique that offers a good compromise between parallel and numerical efficiency. The proposed solver is built around two main components:

- A widely adopted strategy for the SPMD parallelization of finite element type calculations. This strategy maximizes the parallel efficiency of the resulting solver by explicitly enforcing data locality through domain partitioning techniques (see for example [11]). Moreover, by using standard message passing environments such as PVM or MPI, the portability of the solver is also guaranteed.
- A multigrid acceleration technique for the solution of large sparse linear systems arising from linearized implicit time integration techniques or dynamic mesh deformation procedures. A multigrid by volume agglomeration strategy has been selected which has already proven its efficiency on several problems (see for example [1,9,10,18]). Its main advantage relies in the fact that the multigrid hierarchy can be automatically generated using the sole data given by the finest discretization of the computational domain. This aspect is of particular importance in the context of the SPMD parallelization strategy considered in this study: the problem of generating local data structures for coarse grid topologies and data exchange at submesh interfaces is treated in parallel without resorting to an appropriate (multimesh) partitioning technique.

The resulting parallel flow solver has been extensively tested and evaluated in the 2D case. Its application to 3D problems is currently done in the context of the `N3S-NATUR` industrial CFD package; preliminary results have been presented here for steady Euler flows. In addition to the extension of this methodology to steady and unsteady turbulent flows and its application to problems of interest to the involved end-users, an ongoing research work is concerned with the analysis of an additive multigrid variant which is based on a residual/correction filtering technique inspired from the work of Tuminaro [24], which aims at minimizing the degradation of the parallel efficiency when switching the calculation to the coarsest levels.

Acknowledgements

Part of the calculations presented in this paper have been conducted on the `SGI Power Challenge Array` and `SGI Origin 2000` systems installed at the *Centre Charles Hermite* located in Nancy. The `Pentium Pro` cluster is an experimental platform located at INRIA Sophia Antipolis. The authors acknowledge support for the work presented here from *Direction de la Recherche* of RENAULT, *Direction des Etudes et Recherches* of EDF and *Division Recherches et Etudes Avançées* of SNECMA under grant #197E718 – 720/0041606011, and from *Direction des Lanceurs* of CNES under grant #197E660/0041606011.

References

[1] G. Carré, An implicit multigrid method by agglomeration applied to turbulent flows, Computers and Fluids 26 (1997) 299–320.
[2] D. Chargy, N3S-NATUR V1. 2 reference and user manuals, SIMULOG 1998.
[3] C. Farhat, S. Lanteri, Simulation of compressible viscous flows on a variety of MPPs: computational algorithms for unstructured dynamic meshes and performance results, Comp. Meth. Appl. Mech. and Eng. 119 (1994) 35–60.
[4] L. Fezoui, A. Dervieux, Finite element non-oscillatory schemes for compressible flows, Proceedings of the Eighth France–U.S.S.R.–Italy Joint Symposium on Computational Mathematics and Applications, IAN, 730, Pavie, Italy, 1989.
[5] L. Fezoui, B. Stoufflet, A class of implicit upwind schemes for Euler simulations with unstructured meshes, J. Comput. Phys. 84 (1989) 174–206.
[6] W. Hackbusch, Multigrid methods and applications, Springer Series in Computational Mathematics, vol. 4, Springer, Berlin, 1985.
[7] P.-W. Hemker, On the order of prolongations and restrictions in multigrid procedures, J. Comput. Appl. Math. 32 (1990) 423–429.
[8] M. Jaeger, D. Dhatt, An extended $k-\varepsilon$ finite element model, Int. J. Numer. Meth. Fluids 14 (1992) 1325–1345.
[9] B. Koobus, M.-H. Lallemand, A. Dervieux, Unstructured volume-agglomeration MG: solution of the Poisson equation, Int. J. Numer. Meth. Fluids 18 (1994) 27–42.
[10] M.-H. Lallemand, H. Steve, A. Dervieux, Unstructured multigridding by volume agglomeration: current status, Computers and Fluids 21 (1992) 397–433.
[11] S. Lanteri, Parallel solutions of compressible flows using overlapping and non-overlapping mesh partitioning strategies, Parallel Comput. 22 (1996) 943–968.
[12] S. Lanteri, M. Loriot, Large-scale solutions of three-dimensional compressible flows using the parallel N3S-MUSCL solver, Concurrency, Pract. Exp. 8 (1996) 769–798.
[13] B. Larrouturou, How to preserve the mass fraction positivity when computing compressible multicomponent flows, J. Comput. Phys. 95 (1991) 59–84.
[14] B.-E. Launder, D.-B. Spalding, Mathematical Models of Turbulence, Academic Press, London and New York, 1972.
[15] M. Loriot, Mesh Splitter 3D V3. 1 user manual, SIMULOG 1998.
[16] D.-J. Mavriplis, A. Jameson, Multigrid solution of the two-dimensional Euler equations on unstructured meshes, AIAA paper 87-0353, 1987.
[17] D.-J. Mavriplis, A. Jameson, Accurate multigrid solution of the Euler equations on unstructured and adaptive meshes, AIAA paper 88-3707, 1988.
[18] D.-J. Mavriplis, V. Venkatakrishnan, Agglomeration multigrid fot two-dimensional viscous flows, J. Comput. Phys. 24 (1995) 553–570.
[19] K. Mer, Variational analysis of a mixed finite element/finite volume scheme on general triangulations, INRIA Technical Report 2213, 1994.
[20] A.-W. Mulder, A new multigrid approach to convection problems, J. Comput. Phys. 83 (1989) 303–323.
[21] B. Nkonga, H.-G. Guillard, Godunov type method on non-structured meshes for three-dimensional moving boundary problems, Comp. Meth. Appl. Mech. and Eng. 113 (1994) 183–204.
[22] P.L. Roe, Approximate riemann solvers, parameters vectors and difference schemes, J. Comput. Phys. 43 (1981) 357–371.
[23] J.-L. Steger, R.-F. Warming, Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods, J. Comput. Phys. 40 (1981) 263–293.
[24] R.-S. Tuminaro, A highly parallel multigrid-like method for the solution of the Euler equations, SIAM J. Sci. Stat. Comput. 13 (1) (1992) 88–100.
[25] B. van Leer, Towards the ultimate conservative difference scheme V: a second-order sequel to Godunov's method, J. Comput. Phys. 32 (1979) 361–370.
[26] P. Wesseling, An Introduction to Multigrid Methods, Springer Series in Computational Mathematics 4, Pure and Applied Mathematics, Wiley, New York, 1991.