INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2005; **47**:367–385
Published online 29 November 2004 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.804

A multigrid adaptive mesh refinement strategy for 3D aerodynamic design

J.-C. Jouhaud[1,*,†,‡], M. Montagnac[1,§] and L. Tourrette[2,¶]

[1]*European Center for Research and Advanced Training in Scientific Computation (CERFACS), 42, avenue Gaspard Coriolis, 31057 Toulouse Cedex 1, France*
[2]*AIRBUS France, 316, route de Bayonne, 31060 Toulouse Cedex 3, France*

SUMMARY

Presently, improving the accuracy and reducing computational costs are still two major CFD objectives often considered incompatible. This paper proposes to solve this dilemma by developing an adaptive mesh refinement method in order to integrate the 3D Euler and Navier–Stokes equations on structured meshes, where a local multigrid method is used to accelerate convergence for steady compressible flows. The time integration method is a LU-SGS method (*AIAA J* 1988; **26**:1025–1026) associated with a spatial Jameson-type scheme (Numerical solutions of the Euler equations by finite volume methods using Runge–Kutta time-stepping schemes. *AIAA Paper*, 81-1259, 1981). Computations of turbulent flows are handled by the standard k–ω model of Wilcox (*AIAA J* 1994; **32**:247–255). A coarse grid correction, based on composite residuals, has been devised in order to enforce the coupling between the different grid levels and to accelerate the convergence. The efficiency of the method is evaluated on standard 2D and 3D aerodynamic configurations. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: local multigrid; adaptive mesh refinement; structured grids; elsA solver

1. INTRODUCTION

Despite the constant progress in numerical methods and the power of new computers, improving the accuracy and reducing computational costs are still the two major objectives of CFD. These objectives are often considered incompatible. Nevertheless, adaptive mesh

*Correspondence to: J.-C. Jouhaud, European Center for Research and Advanced Training in Scientific Computation (CERFACS), 42, avenue Gaspard Coriolis, 31057 Toulouse Cedex 1, France.
†E-mail: jean-christophe.jouhaud@cerfacs.fr
‡Senior Researcher at CERFACS.
§Research Engineer at CERFACS.
¶Research Engineer at AIRBUS France.

Contract/grant sponsor: Publishing Arts Research Council; contract/grant number: 98-1846389

Received 21 June 2003
Revised 30 April 2004

refinement methods (AMR) seem to overcome this incompatibility by reaching a good compromise between computational costs and accuracy.

Most of the compressible flows are characterized by structures which have different length scales such as shocks, contact discontinuities, viscous layers, vortices, etc. Then, the smallest scale governs the number of grid points in the meshing process and if the whole domain is refined uniformly this can lead to a large number of cells.

Aerodynamicists from AIRBUS France aircraft manufacturer study complex 3D configurations of airplanes (bodies $+$ wings $+$ pylons $+$ nacelles $+\cdots$) with structured meshes. They are penalized by the computational costs associated to the increase of grid points. To avoid this problem, they look towards AMR techniques to locally adapt the mesh size to different structure scales. In fact, if successfully integrated in a production code, such a technique could perform numerical simulations of high quality at a reasonable computational cost.

Nowadays, several techniques allow to locally increase the grid point density thanks to local flow properties. Firstly, the main coarse mesh can be enriched itself thus leading to an unstructured mesh approach [1, 2]. Secondly, in case of numerical methods based on a structured approach, the initial coarse mesh can be enriched with new meshes generated from that basic coarse one. As a consequence, they are nested one in another and are ordered into levels. Each level is built by dividing some cells belonging to meshes of the lower level. Thus, a *Hierarchical Structure of Meshes* results from these successive refinements.

The earliest researches in this field have been realized by Berger and Collela [3] who have first developed cartesian AMR methods for solving 2D unsteady flows in the case of detonic problems. For 2D steady flows, Brandt [4] has developed multi-level adaptive techniques. Recently, an article [5] has introduced a second-order unsplit Godunov method with local time stepping for 2D Euler steady equations as a continuation of the work in Reference [6]. More recently, a 3D AMR method has been applied [7] for the computation of wake vortices. Here, this method was improved in order to accelerate the convergence strategy.

The aim of this work is to develop, implement and assess a multigrid-AMR method in an industrial code. Our contribution consists in adapting the multigrid technique for embedded meshes in the context of the compressible 3D turbulent Navier–Stokes equations for steady transonic flows. This multigrid-AMR method has been developed in the elsA[||] software [8]. The 3D Navier–Stokes equations are solved to steady-state using the second-order centred Jameson scheme [9] and the LU-SGS [10] implicit time integration method, with local time stepping. The automatic grid adaptation process is performed outside the solver by the Mbref utility program developed by AIRBUS France on the basis of the algorithms proposed by Quirk [11].

The paper is organized as follows. In Section 2, the classical governing equations and the elsA solver are introduced. The principles for creating embedded meshes using successive refinements are presented in Section 3. First, the topology chosen for such meshes is exposed introducing the map of interface connectivities. The definition problem of boundary conditions is addressed. Then, the refinement process is described. Section 4 deals with the most important part of this study. The extension of the multigrid technique, as means of coupling the different levels of refinement, is exposed as a 3D generalization of the method introduced by Borrel

[||]French acronym, ensemble logiciel pour la simulation en Aérodynamique, which means aerodynamic simulation software package.

and Jouhaud [12–14]. Section 5 addresses the question of parallel computing in relation with the AMR approach. Indeed, AMR mesh integration on several processors requires an elaborated strategy since some domain interfaces are non-coincident and the distribution of blocks on processors is not trivial. In Section 6, the method is validated on several test-cases representative of aerodynamic problems with a focus on the ratio cost/accuracy.

2. GOVERNING EQUATIONS AND FLOW SOLVER

2.1. Navier–Stokes equations

The governing equations are the 3D Navier–Stokes equations which describe the conservation of mass, momentum and energy of a viscous fluid flow. Using Cartesian co-ordinates, these equations can be expressed in a conservative form as follows:

$$\frac{\partial W}{\partial t} + \nabla \cdot F = 0 \tag{1}$$

The state vector W and the flux $F = F_{\mathrm{e}} - F_{\mathrm{v}}$ decomposed in an inviscid and a viscous part are given by

$$\begin{aligned} W &= [\rho, \rho\mathbf{U}, \rho E]^{\mathrm{T}} \\ F_{\mathrm{e}} &= [\rho\mathbf{U}, \rho\mathbf{U} \otimes \mathbf{U} + p\bar{\bar{I}}, \mathbf{U}(\rho E + p)]^{\mathrm{T}} \\ F_{\mathrm{v}} &= [0, \bar{\bar{\tau}}, \mathbf{U} \cdot \bar{\bar{\tau}} - \mathbf{q}]^{\mathrm{T}} \end{aligned} \tag{2}$$

where ρ is the density, $\mathbf{U}$ the velocity, p the pressure and E the total energy. For a Newtonian fluid, the shear stress tensor $\bar{\bar{\tau}}$ is given by

$$\bar{\bar{\tau}} = \mu(\nabla\mathbf{U} + (\nabla\mathbf{U})^{\mathrm{T}}) + \lambda\nabla \cdot \mathbf{U}\bar{\bar{I}} \tag{3}$$

with μ the dynamic viscosity and λ the second coefficient of viscosity. The Stokes' assumption reduces the Lamé's relation to $2\mu + 3\lambda = 0$. The heat flux q is given by Fourier's law

$$\mathbf{q} = -K_T \nabla T \tag{4}$$

with T the temperature and K_T the thermal conductivity coefficient. The dynamic viscosity μ is given by the Sutherland's formulae

$$\mu = \mu_0 \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + C_s}{T + C_s} \tag{5}$$

where μ_0 is the dynamic viscosity at the reference temperature T_0 and the constant C_s equals to 110.3 K. With a constant Prandtl number, the heat conductivity can be written as $K_T = \mu C_p / Pr$ with C_p the specific heat at constant pressure and $Pr = 0.72$ for air. For a caloric perfect gas, the state equation is given by $p = \rho RT$ where the gas constant R is equal to 287 (J/kg K) for air.

2.2. *Wilcox k–w turbulence model*

The k–ω model equations are given by system (6)

$$\begin{aligned}
&\frac{\partial \rho k}{\partial t} + \nabla \cdot (\rho k \mathbf{U}) - \nabla \cdot ((\mu + \sigma_k \mu_t)\nabla k) = P_k - \beta_k \rho \omega k \\
&\frac{\partial \rho \omega}{\partial t} + \nabla \cdot (\rho \omega \mathbf{U}) - \nabla \cdot ((\mu + \sigma_\omega \mu_t)\nabla \omega) = \frac{\alpha_\omega \omega}{k} P_k - \beta_\omega \rho \omega^2
\end{aligned} \tag{6}$$

with the production term $P_k = \tau \cdot \nabla \mathbf{U}$, the eddy viscosity $\mu_t = \rho k/\omega$, the closure coefficients $\sigma_k = 0.5$, $\sigma_\omega = 0.5$, $\beta_k = 0.09$, $\beta_\omega = 0.075$, $\alpha_\omega = 0.5$, k the turbulent kinetic energy and ω the specific turbulent dissipation.

2.3. *Flow solver*

The elsA software [8] is the platform used in the present study. It has been initiated at ONERA and aims at gathering a very large range of CFD capabilities in a software package to tackle industrial problems as well as becoming a platform for further innovative CFD developments. It is included in an object-oriented framework to ease software management and is operational in the AIRBUS production environment and in the CERFACS research environment since the beginning of year 2004.The elsA code solves the 3D turbulent compressible Navier–Stokes equations using a cell-centred finite-volume method.

Integrating Equation (1) over a domain Ω and applying Green's divergence theorem yield the following integral form:

$$\int_\Omega \frac{\partial U}{\partial t}\,\mathrm{d}V + \oint_{\partial\Omega} F \cdot \mathbf{n}\,\mathrm{d}S = 0 \tag{7}$$

with $\mathbf{n}$ the outward normal of the boundary $\partial\Omega$ of the control volume Ω. The separated time/space discretization process leads to the following delta form:

$$A\Delta U^{n+1} = -\frac{\Delta t}{|\Omega|} R(U^n) \tag{8}$$

where the residual R comes from the space discretization and depends on the conservative variable field U^n. The Jacobian matrix A comes from the implicit time discretization and $\Delta U^{n+1} = U^{n+1} - U^n$ corresponds to the field correction also called the time increment.

3. HIERARCHICAL STRUCTURE AND BOUNDARY CONDITIONS

3.1. *Hierarchical structure definition*

The AMR method is based on a physical space discretization of relevant flow regions thanks to the several meshes built by successive refinements. At the beginning of the process, only the coarsest mesh is given by the user. It is called the *basic mesh* G_0 since it remains unchanged upon all computation. Most of the time, this mesh is a very rough discretization of the entire problem domain. Nevertheless, it should allow a first approximation of the solution in order to start the refinement process. Then, different parts of the basic mesh are refined depending on some criteria based on the flow solution. The same area might be successively refined several times. Once a level has been created, then it cannot be altered by the next refinement step. In the following, cells are divided by two in each space directions. Thus, the computational domain becomes more and more refined and this set of embedded meshes is called a *hierarchical structure of grids*. If the present refinement level is noted l and the corresponding grid G_l, the hierarchical structure G can be defined as follows:

$$G = \bigcup_{l=0}^{l=l_{\max}} G_l \quad \text{with } G_l = \cup_i \; G_{l,i} \text{ where } G_{l,i} \text{ are blocks}$$

3.2. Topological properties

To ensure a coherence, a mesh structure G (cf. Figure 1) must respect some fundamental topological rules that can be sum up by the *Properly Nested properties* [14]:

1. Inclusion of different levels: $G_l \subset G_{l-1}, \ \forall l \ 1 \leqslant l \leqslant l_{\max}$
2. Non-overlapping of blocks belonging to a same level: $G_{l,i} \cap G_{l,j} = \emptyset$ when $i \neq j$
3. Cells from a block $G_{l,i}$ of the set G_l, which are next to a border interface excluding physical boundary conditions, must not have adjacent cells belonging to a set G_k with $l+1 < k$ or $k < l-1$.

The first rule ensures the successive nesting of grids, i.e. the gradual basic mesh enrichment with finer meshes.

The second rule is a standard choice that prevents grids from overlapping, which is very expensive in computation time and memory storage. However, its not the only choice possible, in particular, when working with rotating embedded meshes [3].

The third rule leads to a gradual distribution of refinement zones from the coarsest to the finest. The difference between the refinement levels at border interfaces of blocks should not be greater than one. It avoids numerical instabilities and accuracy issues by preventing too

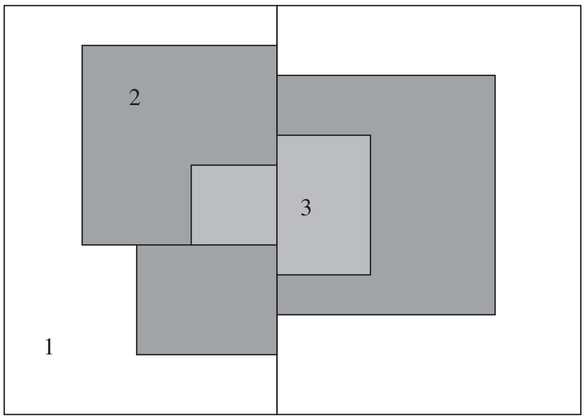

Figure 1. Properly Nested hierarchical structure.

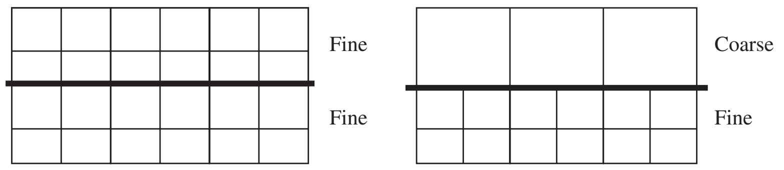

Figure 2. Fine/fine and fine/coarse interfaces.

large difference in cell sizes at interfaces between two blocks of different level [15]. However, it permits a finer and finer refinement near boundary conditions.

3.3. *Block boundary conditions*

The boundary conditions of blocks are implemented thanks to two rows of ghost cells. This is sufficient for the fourth-order term of the scalar artificial dissipation scheme used. This classical treatment allows to generalize the flux calculation at all cell interfaces.

Values in ghost cells are calculated at each iteration from interior cells of the neighbouring blocks. Due to the *Properly Nested* properties, three kinds of block border interfaces can appear:

- a classical physical boundary condition,
- a fine/fine interface connection,
- a fine/coarse interface connection.

The fine/fine interface connections are coincident connections (cf. Figure 2) between two blocks at the same level of refinement. Mesh lines are continuous across the interface. In this case, ghost cells of one block are filled with the corresponding opposite cell values (interior cells of the neighbouring block).

The fine/coarse interface connections (cf. Figure 2) are partially coincident connections between two blocks at a different level of refinement. One mesh line over n is continuous across the interface. A trilinear interpolation is used to set the ghost cell values of the fine block from the overlapped cell values of the coarse block.

3.4. *Automatic block construction*

The mesh adaptation is performed in order to refine regions of interest (shocks, boundary layers, etc.). AIRBUS France has developed a suite called GAME** which includes an automatic mesh adaptation tool named Mbref and a CFD solver. This suite is able to automatically refine the mesh from the first results obtained on the basic coarse mesh. The Mbref tool adapts the mesh to the flow solution by introducing blocks in relevant regions detected by flow sensors (cf. Figure 3). It also projects new nodes on CAD surfaces when necessary.

In fact, the Mbref tool offers several sensors based on numerical or physical criteria. For instance, in a transonic flow context, the following physical sensor is used to detect shocks

**Grid adaptation by mesh enrichment.

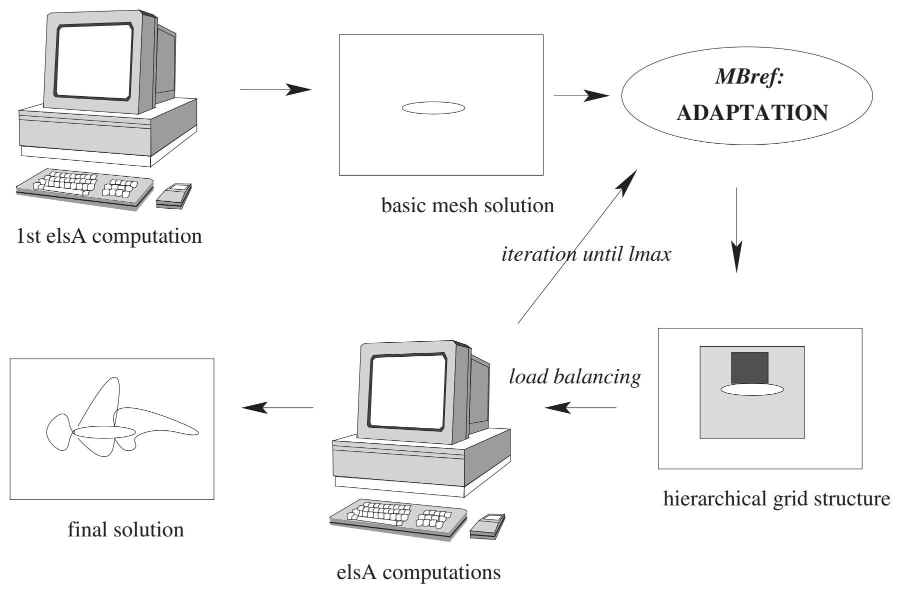

Figure 3. GAME computational suite.

and compression zones at leading edge. It is based on an estimation of the density gradient using the α coefficient [11] for each cell

$$\alpha = \frac{\|\overrightarrow{\mathrm{grad}}(\rho)\|}{\max(\|\overrightarrow{\mathrm{grad}}(\rho)\|)}$$

After the first computation on the basic mesh G_0, α is calculated for all cells. If its value is greater than a critical α_0 value then cells are flagged to be refined. Then, the smallest block that could contain all cells is determined. If the ratio of the number of flagged cells over the total number of cells in this block is greater than a defined value, then it is created, otherwise the block is divided by two along the largest topological direction. Then, the process is repeated for the two blocks until convergence. This form-recognizing process is called *Grouping/Clustering Algorithm* and it has been fully described by Quirk [11].

4. MULTIGRID AND AMR

The classical AMR method, i.e. the method developed by Berger and Collela [3] or Quirk [11], is valid only for unsteady flows. So, its no more valid for steady flows because the convergence is not guaranteed. In case of 2D steady flows, Dannenhoffer and Baron [16] have tried to combine AMR and multigrid methods. In fact, multigrid methods aim at coupling and accelerating different grid resolutions for steady flows. Then, this is a good choice to adapt them for local mesh refinement.

In the following, a 3D local multigrid technique, which represents an extension from Brandt's standard full approximate storage (FAS) algorithm [4, 17] is proposed. Based on a multigrid approach, this technique [12–14] couples the different AMR levels in order to accelerate the convergence to steady state.

4.1. *Description of the local multigrid approach* (*local forcing function*)

The following description is based on Jameson's formulation [6, 18] (formulation of Brandt's standard FAS for compressible flows) where the coarse grid solution is driven by the residual computed on the fine grid, through a residual forcing function.

Contrary to the standard case where all grids cover the whole computational domain, in our case, the portion of physical space covered by successively finer grids $G_1, \ldots, G_{l\max}$ gets reduced. Thus, the classical restriction and prolongation operators need to be redefined. In order to link the multigrid method with the AMR method, a *local forcing function* is introduced to allow a local formulation of Jameson's FAS. This local forcing function modifies coarse block residuals by constructing the so-called *composite residuals*.

Let us consider two properly nested grids G_l and G_{l-1}. The equation to solve on the finest grid G_l can be written in a delta-form (8) as follows:

$$A_l \Delta U_l^{n+1} = -\frac{\Delta t_l}{|\Omega_l|} R_l(U_l^n) \tag{9}$$

with $\Delta U_l^{n+1} = U_l^{n+1} - U_l^n$.

(a) Equation (9) is first solved. Starting with the increment ΔU_l^{n+1}, a new solution is computed: $U_l^{n+1} = \Delta U_l^{n+1} + U_l^n$ and the new residual $R_l(U_l^{n+1})$ is calculated. Then U_l^{n+1} and $R_l(U_l^{n+1})$ are locally restricted on the grid G_{l-1} thanks to a local conservative restriction operator T_l^{l-1} from level 1 to level 1-1 as

$$U_{l-1\ |M_{l-1}}^{n+1} = T_l^{l-1} U_l^{n+1} \tag{10}$$

$$R_{l-1}^c = T_l^{l-1} R_l(U_l^{n+1}) \tag{11}$$

where M_{l-1} is the subset of cells from G_{l-1} that have been refined. The main difference with the classical FAS algorithm is that the projection of U_l^{n+1} and $R_l(U_l^{n+1})$ is only local. Thus, R_{l-1}^c is only defined on M_{l-1} and $T_l^{l-1} U_l^{n+1}$ is only a part of the present solution on G_{l-1} that can be called U_{l-1}^*.

(b) For the grid G_{l-1}, a local forcing function is defined on M_{l-1} as

$$C_{l-1} = R_{l-1}^c - R_{l-1\ |M_{l-1}} = T_l^{l-1} R_l(U_l^{n+1}) - R_{l-1}(U_{l-1}^{n+1})_{|M_{l-1}} \tag{12}$$

Then, a *composite residual* $R_{l-1}^{\text{comp}}(U_{l-1}^*)$ has to be built on G_{l-1}. This residual has two definitions depending on whether it is built on refined cells or not:

$$\textit{Refined cells} \quad R_{l-1\ |M_{l-1}}^{\text{comp}} = R_{l-1}(U_{l-1}^*)_{|M_{l-1}} + C_{l-1} \tag{13}$$

$$\textit{Non-refined cells} \quad R_{l-1\ |(G_{l-1}\backslash M_{l-1})}^{\text{comp}} = R_{l-1}(U_{l-1}^*)_{|(G_{l-1}\backslash M_{l-1})} \tag{14}$$

Equation (13) is the classical FAS formulation. By now, the composite residual of the grid G_{l-1} is defined on every cell. So Equation (15) has to be solved on level $l-1$

$$A_{l-1}\Delta U_{l-1}^{n+2} = -\frac{\Delta t_{l-1}}{|\Omega_{l-1}|} R_{l-1}^{\mathrm{comp}} \tag{15}$$

Its important to notice that the composite residual leads to the conservation of physical quantities between grid levels. In fact, this composite residual is composed of terms only depending on residual formulations: restriction of fine residuals or computation of local residuals. Furthermore, a refluxing procedure permits to perform a flux balance at all fine/coarse interfaces. It consists in imposing the fine–coarse interface fluxes (coarse cell part) by summing the fine fluxes (fine cell part).

(c) The last step is the prolongation from coarse to fine grids. It is a local implementation of the traditional way up in the V-algorithm, i.e. the local prolongation P_l^{l-1} from coarse level $l-1$ to fine level l

$$U_{l-1}^{n+2} = \Delta U_{l-1}^{n+2} + U_{l-1}^{n+1} \tag{16}$$

$$U_l^{n+2} = U_l^{n+1} + P_{l-1}^{l}(U_{l-1\,|M_{l-1}}^{n+2} - U_{l-1\,|M_{l-1}}^{n+1}) \tag{17}$$

In the following, steps (a)+(b) will be called the AMR1 method, and the AMR1 method with step (c) will be called the AMR2 method. The difference between the AMR1 and AMR2 methods has been introduced to follow the implementation progress. Indeed, the AMR1 method does not contain prolongation steps. For that reason, this method could not be considered as a multigrid method but in the case of local mesh refinement, it is the first step to couple the resolution between levels as shown in Reference [7]. It should be emphasized that the AMR1 and AMR2 methods converge towards the same solution. The only difference concerns the convergence rate as shown in Section 6.

This multigrid AMR method can be generalized to every number of level and every kind of cycles (V,W, etc.). The restriction operator is a simple volume-weighted average. The prolongation operator is a classical node-to-cell trilinear interpolation [8].

4.2. Local multigrid, global multigrid (GMG) and AMR framework

In this paper, a difference is made between the local multigrid (AMR framework) and the standard global multigrid (GMG) where the coarse meshes generated from the basic mesh cover the entire computational domain.

The AMR+GMG computation consists first in computing a solution on the basic coarse grid G_0 (with the standard global multigrid) without any refinement. As long as characteristic flow structures do not appear on G_0 or are not stabilized, there is no reason to refine: sensors could needlessly refine unimportant regions. This is also why the basic coarse mesh must be carefully defined in order to capture the viscous effects in turbulent configurations. However, a full convergence does not necessarily need to be reached since the coarse grid solution will be modified by the coupling with refinement levels.

After that, for each new level created (Mbref tool), the initial solution is defined by prolongating the solution of the last coarser level (cf. Figure 4). Then, the AMR+GMG

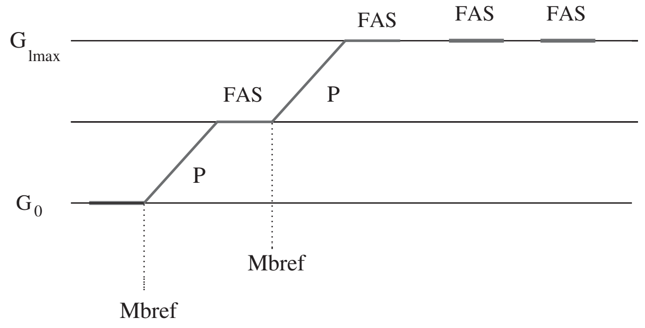

Figure 4. AMR multigrid cycle definition.

method is applied between the finest grid and the coarsest grid (coarsening of G_0). This is repeated until the finest refinement is reached.

In the following, an AMR computation do not take into account a coarsening of the basic grid G_0 but only the refined levels.

5. PARALLELIZATION OF THE LOCAL MULTIGRID AMR METHOD

The code parallelization is a key point for industrial applications. Much attention has to be paid to get good speed-ups. One of the main point for the local multigrid AMR approach is to distribute the blocks according to their levels otherwise this may cost a lot of CPU time and be very complex to implement.

In addition to the properly nested properties two further rules have been imposed to make the implementation of parallelization easier and efficient:

- the first one specifies that a block $G_{l,i}$ cannot overlap two blocks of level $l-1$. It must be entirely included in the block from which it has been generated: $G_{l,i} \subseteq G_{l-1,j}$. This rule is also applied sequentially to ensure consistency in the code implementation.
- the second rule imposes the constraint that each block must be placed on the same processor than the block in which it is included. In other words, a block must always stay with its 'relative'.

As a consequence these two rules enable to avoid exchanging huge amount of volumic data between processors during the prolongation and restriction steps. Thus, only surfacic data are exchanged between processors to update ghost cells at the border interfaces of the blocks.

On the other hand, the load balancing could deteriorate. In fact, as these rules are also included in the load-balancing tool this aspect is not really important for complex geometries that contain many blocks. In our cases which include only two basic coarse meshes, this is the worst case that can occur since only two processors can be used following the two previous rules. When the fine levels are added to the initial configurations then the load-balancing becomes imperfect. These are the reasons why no speed-up figures will be presented.

Nevertheless it should be noticed that the performances on AMR configurations are expected to be the same as on other AMR-free configurations as long as the load-balancing is excluded from the analysis. No specific parallel operations are done by the AMR implementation thanks to the rules introduced.

6. NUMERICAL RESULTS

In this section, the local multigrid AMR method has been validated on standard configurations. The following test cases have been considered: a 2D Euler transonic flow around a NACA0012 airfoil, a 3D Euler transonic flow around an AS28 wing and a 2D turbulent flow around a RAE2822 airfoil. In particular, our attention was focused on the CPU time savings and the accuracy. For all these computations, the classical second-order central scheme of Jameson [6] is used with the LU-SGS time implicit integration method [10] as the smoother on the fine meshes and as the solver on the coarsest meshes.

A 5-point V-algorithm has been chosen for the local/global multigrid since all computations were performed with two levels of refinement. Two iterations are performed on all grids except on the finest one. Refinements are always isotropic that is to say each coarse cell is divided by two in each space direction to give eight fine cells in 3D.

All computations have been performed with the parallel mode on with two processors in order to test the good behaviour of the AMR functionalities. Nevertheless, to compare precisely the CPU times the results presented in the paper are the ones of a sequential version.

The residuals showed on the figures are computed by taking into account all the meshes except these coming from the global multigrid (coarsening of the basic mesh).

6.1. Transonic Euler flow around a NACA0012 airfoil

The transonic Euler flow around the NACA0012 airfoil is considered with the Mach number $M_\infty = 0.85$ and the angle of attack $\alpha = 1^\circ$. The flow has two supersonic zones with a strong shock on the upper surface and a weaker one on the lower surface. The accuracy of the solution is evaluated on the basis of the pressure coefficient distribution.

The initial basic mesh is made of two structured C-mesh containing 108×38 cells each, which represents a quite coarse space discretization and the final composite mesh (see Figure 5) with two levels of refinement contains 24 620 more cells. Computations have been performed on a Compaq alpha-server computer.

The solutions obtained on the basic coarse mesh and the two fine levels are shown in Figure 6. The improvement in the prediction of the shock thickness can be clearly seen. This is confirmed by the distribution of the pressure coefficient plotted in Figure 8.

The convergence histories in Figure 7 show the better convergence rate of the AMR2 compared to the AMR1 due to the effect of the prolongation. The addition of a coarsening of the initial basic mesh (AMR+GMG) is even more efficient. The evolution of the lift and drag coefficients is plotted in Figure 9 which shows that they converge faster to their final values for the AMR2 compared to the AMR1, and even faster if a coarse global multigrid mesh (AMR+GMG) is added.

6.2. Transonic Euler flow around an AS28 wing

The transonic Euler flow around an AS28 wing designed by AIRBUS France is considered with the Mach number $M_\infty = 0.80$ and the angle of incidence $\alpha = 2.2^\circ$. Three different meshes

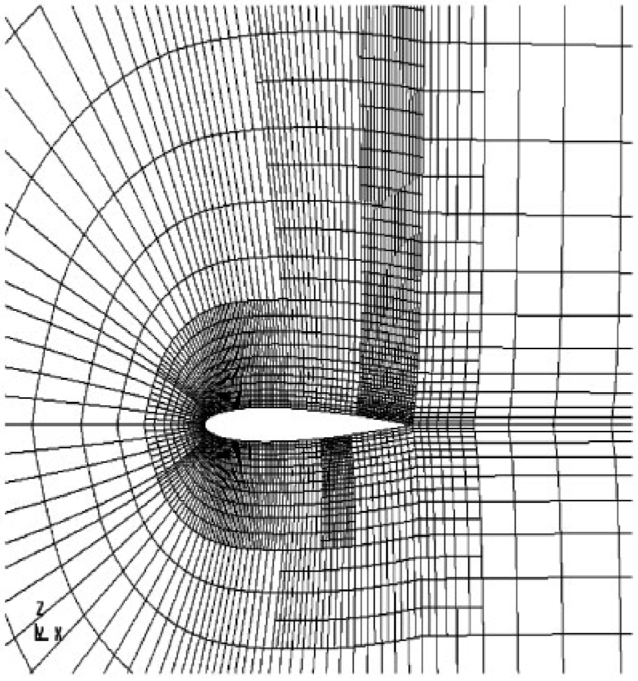

Figure 5. NACA0012, two-level mesh hierarchy.

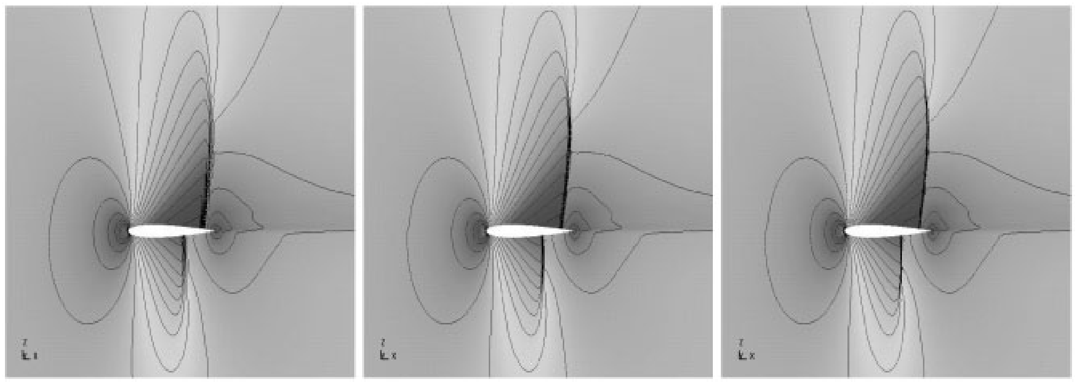

Figure 6. NACA0012, Mach number contours on levels 0, 1 and 2.

are considered for this configuration and showed in Figure 10: an initial coarse mesh containing 28 000 points, a composite mesh with the basic mesh and two levels of refinement containing 502 084 nodes and a globally refined mesh which contains 1 626 690 nodes. Computations have been performed on the Fujitsu VPP5000 supercomputer of Météo France.

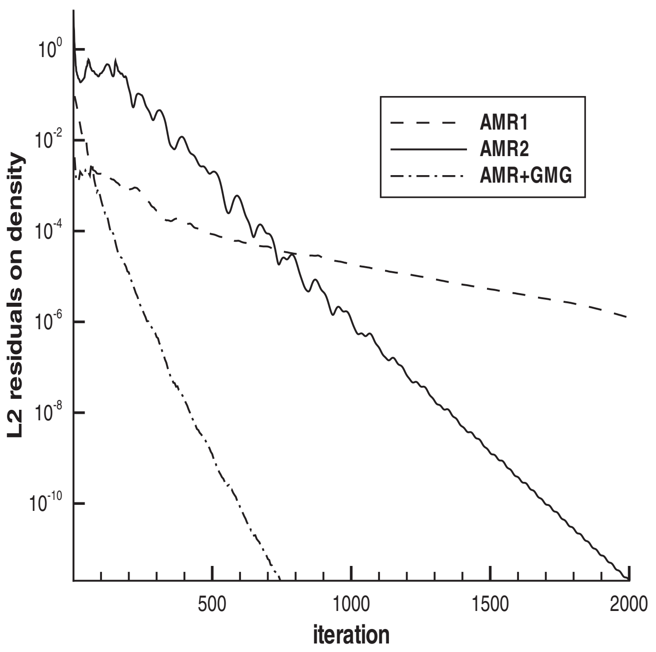

Figure 7. NACA0012, convergence histories.

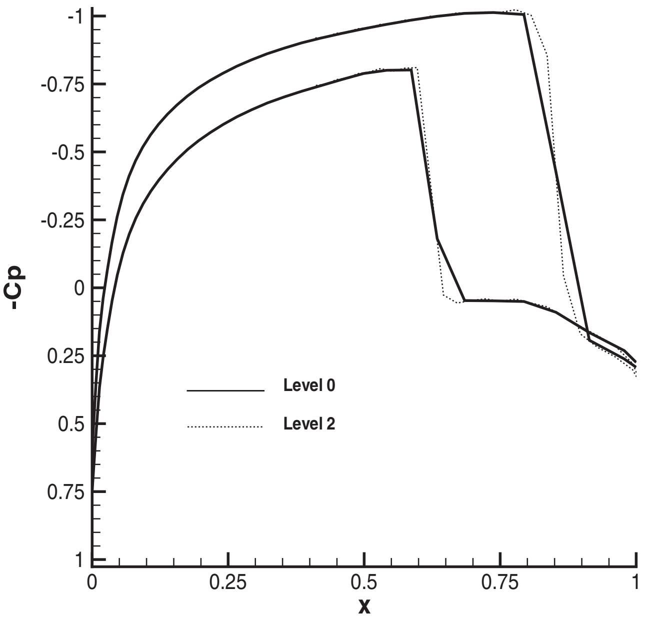

Figure 8. NACA0012, -Cp stations.

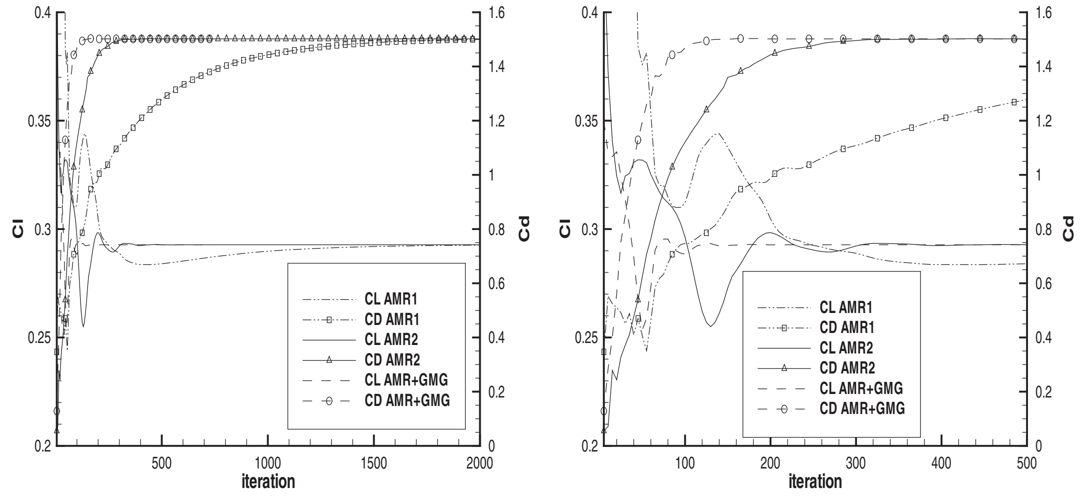

Figure 9. NACA0012, evolution of the lift and drag coefficients.

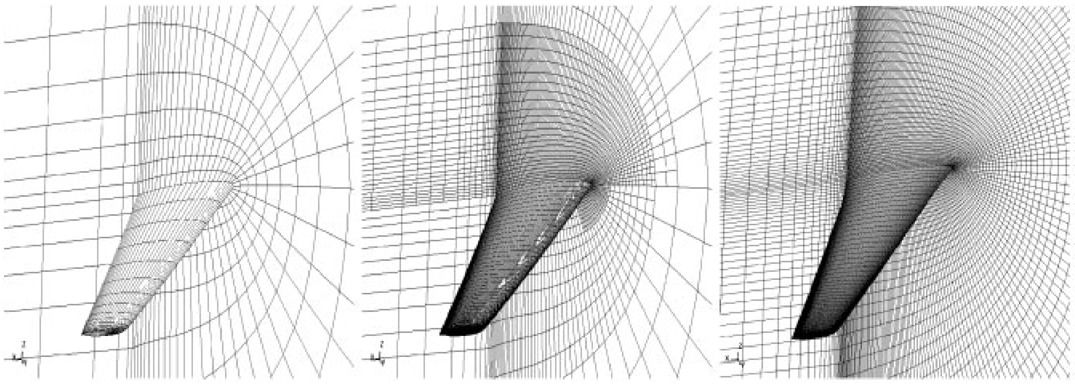

Figure 10. AS28, from left to right: the basic coarse mesh, the two-level AMR mesh first refinement and the globally refined mesh.

Figure 11 represents the Mach number contours on the basic mesh and the different levels of refinement for the AMR computation. It shows the better representation of the shock on the finest mesh.

An important point is to know whether the accuracy of the multigrid AMR solution is closed to the fine case or not. Figure 12 points out the slight differences between the two-level AMR mesh solution and the globally refined mesh solution. The results are quasi-similar except for the shock location, which is better observed with the wall distributions of pressure coefficients for sections $x = 8$ and 13 in Figure 14. The shock location predicted with the two-level AMR method slightly differs from the one predicted with the fine grid. This behaviour has already been explained in the past [12]. It usually corresponds to the strong influence of

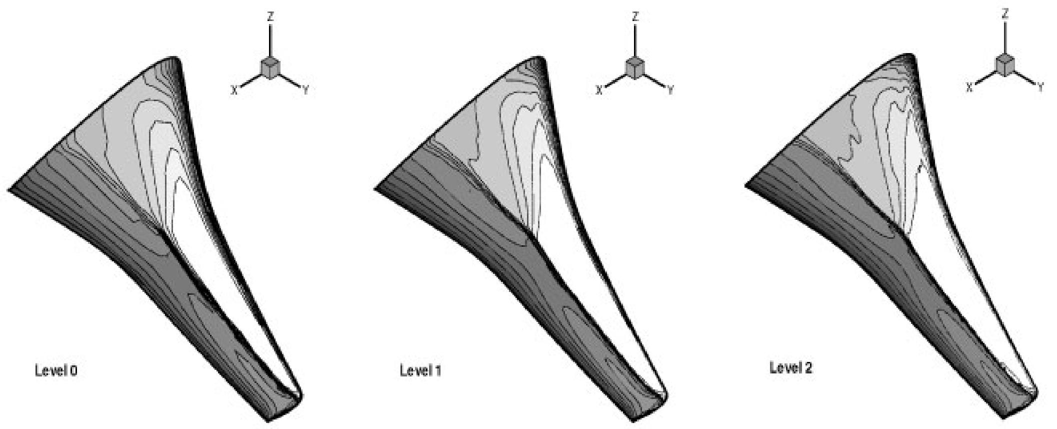

Figure 11. AS28, Mach number contours on the basic mesh and on the two levels.

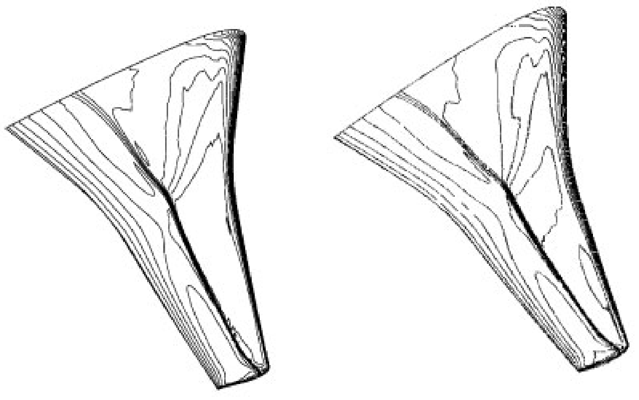

Figure 12. AS28, Mach number contours, differences between the two-level AMR mesh (left part) and the globally refined mesh (right part).

the farfield boundary conditions in transonic flows. In fact the globally refined mesh offers 64 times more points of discretization at the farfield boundary conditions than the AMR mesh since there are no refinements at these boundaries (see Figure 10). This explains the different shock locations.

The convergence rates are presented in Figure 13 for the AMR (*AMR*), for the basic coarse mesh (*coarse*) and for the globally refined mesh (*fine*). It can be noticed that they are similar for the coarse mesh and the AMR mesh. The space discretization being the same at farfield boundary conditions for the AMR and the coarse meshes, it is not surprising to observe the same convergence rate. Since the influence of the farfield boundary conditions is strong in transonic flows, the coarse mesh seems to pilot the convergence of the whole computation.

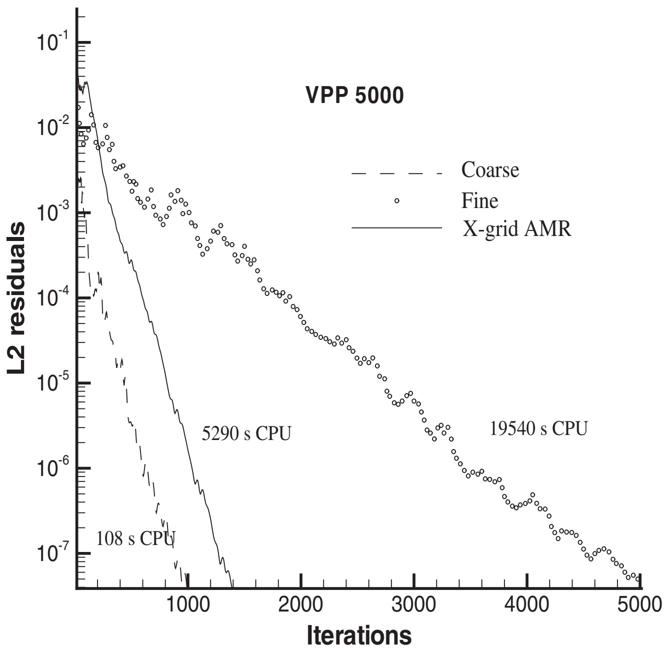

Figure 13. AS28, convergence histories.

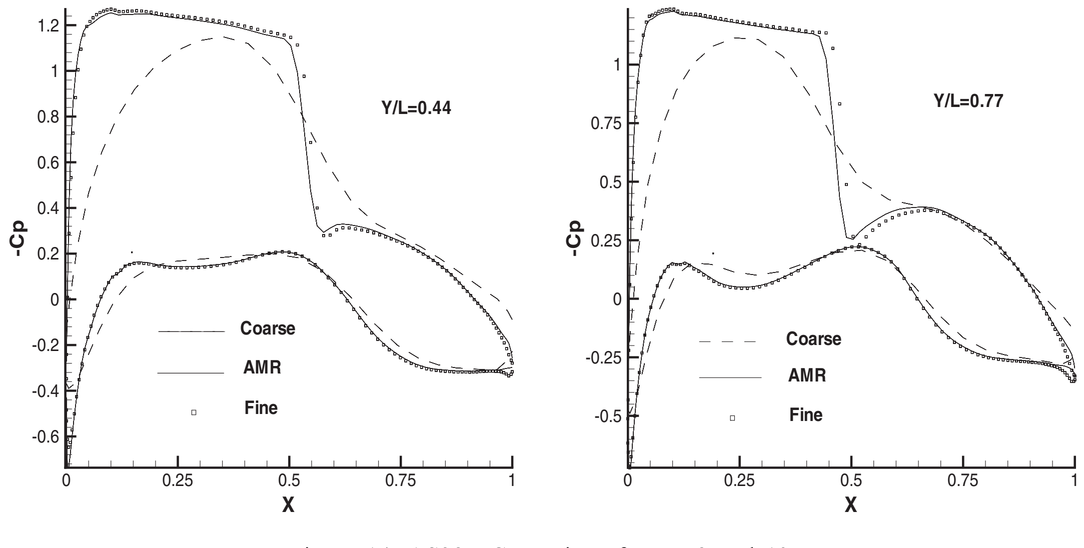

Figure 14. AS28, -Cp stations for $x = 8$ and 13.

Concerning CPU times, the globally refined mesh takes 19 540 s to converge against 5290 s for AMR, i.e. a factor of 3,6 time saving.

This case is representative of the aerodynamical configuration usually encountered. In particular, the refinement rate is typical of such configurations. Then it can be supposed that the above CPU time results can be extrapolated to other configurations.

It can be predicted that for a mesh of 1 000 000 points, a standard computing without refinement strategies takes approximately 3.8 s per iteration whereas the AMR strategy takes around 7.5 s per iteration. This means that the CPU time overhead of the AMR method is nearly of a factor 2. But the number of points needed is around three times less and above all the convergence rate is improved.

Furthermore a decrease of a factor 2.5 in the memory size requirement of the AMR method has been noticed compared to the global refined case. This factor is close to the factor 3 of the point reduction.

6.3. Transonic turbulent flow around a RAE2822 airfoil

The RAE2822 airfoil turbulent configuration has been considered with the k–ω turbulence model. This is a transonic flow at $M_\infty = 0.73$ and 2.79° angle of attack. This flow is characterized by a supersonic zone with a shock on the upper surface. As for the NACA0012 airfoil, the solution accuracy is measured through the distribution of the pressure coefficient. Calculations were run on a SGI Origin2000 computer.

The initial basic mesh has 183 616 cells, which is already a quite fine space discretization. Indeed the basic mesh takes into account the viscous effects to get a first approximation of the solution in order to start the refinement process. Figure 15 shows the mesh as well as the Mach number contours and Figure 16 shows the convergence history of a two-level computation. Pressure coefficient distributions on the airfoil are plotted for all grid levels in Figure 16. It can be noticed that computations are in good agreement with experimental data with only few discrepancies. The shock is clearly better represented on the finest level 2 due to the local mesh refinement. This case allows the validation of the multigrid AMR method for Reynolds average Navier–Stokes equations.

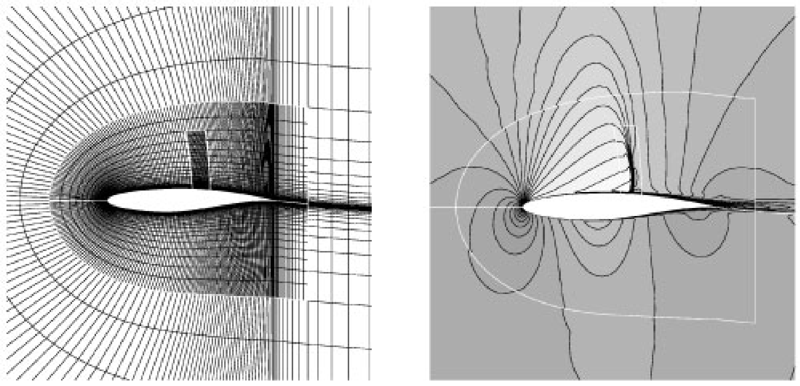

Figure 15. RAE2822, the two-level mesh and Mach number contours.

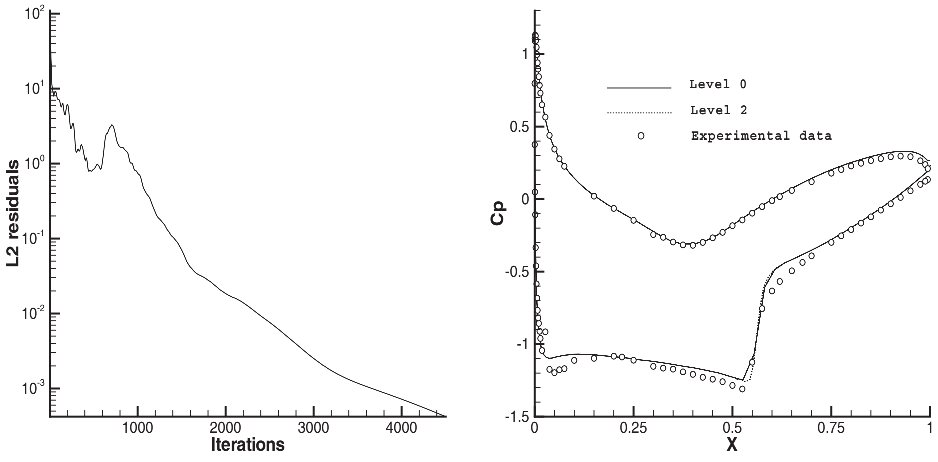

Figure 16. RAE2822, convergence history and -Cp stations.

7. CONCLUSION

In this paper, a local multigrid method combined with an AMR strategy has been developed to compute steady solutions in the elsA software. The convergence was accelerated with local time stepping. A Jameson's scheme has been used for spatial discretization and a LU-SGS implicit method for the time integration. The k–ω model of Wilcox has been implemented to compute turbulent flows in an AMR framework but the extension to other turbulence models is straightforward.

Three test cases were run for external transonic flows over NACA0012 and RAE2822 airfoils, and an AS28 wing. The results show that the AMR method significantly reduces the number of points needed and that it preserves nearly the same accuracy as with the fine mesh. Therefore, computational cost and memory storage are reduced, which is very promising for the intensive use of CFD in aircraft design at AIRBUS France.

The refinement process should be improved to enable a more flexible mesh generation and more intensive testing should be made in parallel to assess the efficiency of the implementation on complex configurations. Other turbulence models should be tested, also with a coarser initial mesh than the one used in the RAE example.

AMR+GMG (cf. 4.2) performs well on Euler equations. It has still to be improved for viscous flows since the turbulence equations are currently solved on all levels but it would probably be better to only restrict the viscosities (no treatments of turbulence equations) on the coarse grids generated from the basic mesh.

It should be kept in mind that multigrid AMR is efficient to solve multi-equations since equations can easily be switched depending on the refinement level or the block locations. So different physical models could be applied depending on the grid levels. Furthermore, different numerical schemes could be combined in order to capture local structures better (for example, high-order schemes only on the finest grid).

Another point is to study the extension of this method to multilevel-based scale-similarity model for LES simulation [19].

REFERENCES

1. Aftosmis MJ. An adaptive grid embedding in non-equilibrium hypersonic flows. *AIAA Paper 89-1652*, 1989.
2. Powell KG, De Zeeuw D. An adaptively-refined cartesian mesh solver for Euler equations. *AIAA Paper 91-1542*, 1991.
3. Berger MJ, Collela P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 1989; **82**:67–84.
4. Brandt A. Multi-level adaptive techniques. IBM I.J. Watson Research Center, Yorktown Heights, New York. *IBM Research Report RC6026*, 1976.
5. Dudek SA, Colella P. Steady-state solution-adaptive Euler computations on structured grids. *AIAA Paper 98-0543*, 1998.
6. Berger M, Jameson A. Automatic adaptive grid refinement for the Euler equations. *AIAA Journal* 1985; **23**(4):561–568.
7. Benkenida A, Bohbot J, Jouhaud JC. Patched grid and adaptive mesh refinement strategies for the calculation of the transport of vortices. *International Journal for Numerical Methods in Fluids* 2002; **40**:855–873.
8. Cambier L, Gazaix M. elsA: an efficient object-oriented solution to CFD complexity. *40th AIAA Aerospace Science Meeting and Exhibit*, Reno, 14–17 January 2002, *AIAA Paper 2002-0108*.
9. Jameson A, Schmidt W, Turkel E. Numerical solutions of the Euler equations by finite volume methods using Runge–Kutta time-stepping schemes. *AIAA Paper 81-1259*, 1981.
10. Yoon S, Jameson A. Lower-upper symmetric–Gauss–Seidel method for the Euler and Navier–Stokes equations. *AIAA Journal* 1988; **26**(9):1025–1026.
11. Quirk JJ. An adaptive grid algorithm for computational shock hydrodynamics. *Ph.D. Thesis*, Cranfield Institute of Technology, College of Aeronautics, 1991.
12. Jouhaud JC, Borrel M. A hierarchical adaptive mesh refinement method: application to 2D flow. *ECCOMAS CFD Conference*, Paris, September 1996.
13. Jouhaud JC, Borrel M. Discontinuous Galerkin and MUSCL strategies for an adaptive mesh refinement method. *15th International Conference on Numerical Methods in Fluids Dynamics*, Monterey, CA, June 1996.
14. Jouhaud JC. Méthode d'adaptation de maillages structurés par enrichissement. *Ph.D. Thesis*, Université Bordeaux I, 1997.
15. Babuska I, Rheinboldt W. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis* 1978; **4**(15):736–754.
16. Dannenhoffer JF, Baron JR. Grid adaptation for the 2D Euler equations. *AIAA Paper 85-0484*, 1985.
17. Brandt A. Multi-level adaptive solutions to boundary value problems. *Mathematics of Computation* 1977; **21**:333–390.
18. Jameson A. Steady state solutions of the Euler equations for transonic flow by a multigrid method. *Advances in Scientific Computation*. Academic Press: New York, 1982; 37–70.
19. Terracol M, Sagaut P, Basdevant C, Gleize V. Une méthode multiniveau pour la simulation des grandes échelles des écoulements turbulents compressibles. *Comptes Rendus de l' Académie des Sciences*, *Série II* 2000; **328**: 81–86.
20. Bohbot J, Jouhaud JC, Darracq D. Coupled patched grid/AMR meshing technics for aircraft design. *40th AIAA Conference*, Reno, January 2002, *AIAA Paper 2002-0546*.
21. Ducros F, Laporte F, Soulères Th, Guinot V, Moinat Ph, Caruelle B. High-order fluxes for conservatives skew-symmetric-like scheme in structured meshes: application to compressible flows. *Journal of Computational Physics* 2000; **161**:114–139.
22. Wilcox DC. Simulation of transition with a two-equation turbulence model. *AIAA Journal* 1994; **32**(2):247–255.