INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2010; **62**:994–1012
Published online 14 July 2009 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.2058

Multigrid convergence acceleration for implicit and explicit solution of Euler equations on unstructured grids

Ali Ramezani and Karim Mazaheri*,†

Department of Aerospace Engineering, Sharif University of Technology, Azadi Street, P.O. Box 11365-8639, Tehran, Iran

SUMMARY

The multigrid method is one of the most efficient techniques for convergence acceleration of iterative methods. In this method, a grid coarsening algorithm is required. Here, an agglomeration scheme is introduced, which is applicable in both cell-center and cell-vertex 2 and 3D discretizations. A new implicit formulation is presented, which results in better computation efficiency, when added to the multigrid scheme. A few simple procedures are also proposed and applied to provide even higher convergence acceleration. The Euler equations are solved on an unstructured grid around standard transonic configurations to validate the algorithm and to assess its superiority to conventional explicit agglomeration schemes. The scheme is applied to 2 and 3D test cases using both cell-center and cell-vertex discretizations. Copyright © 2009 John Wiley & Sons, Ltd.

Received 16 October 2008; Revised 26 February 2009; Accepted 2 March 2009

KEY WORDS: multigrid; implicit; agglomeration; compressible flows; unstructured grids; cell-vertex; cell-center

*Correspondence to: Karim Mazaheri, Department of Aerospace Engineering, Sharif University of Technology, Azadi Street, P.O. Box 11365-8639, Tehran, Iran.
†E-mail: mazaheri@sharif.edu

Contract/grant sponsor: Sharif University of Technology

1. INTRODUCTION

In spite of all the progress in algorithm development, many impediments still need to be overcome before computational fluid dynamic (CFD) techniques can be routinely employed for the design of aerospace vehicles, which would require robust, accurate and efficient computations of intricate flow fields. The resolution of flow phenomena to an appropriate extent for a large set of design and off-design parameters involves computational time scales, which are prohibitive even with the high-performance computers available at present. This problem gets compounded when the conservation equations for mass, momentum and energy have to be supplemented by additional partial differential equations representing complex physical phenomena that may include compressible turbulence and chemical reactions among others. These considerations necessitate the optimization of algorithms for Euler and Navier–Stokes computations to a high degree. In particular, the role of convergence acceleration is of paramount importance in CFD applications.

Numerical solution of Euler equations is still improving in convergence speed and accuracy. After fast growth of unstructured grid usage for complex geometry applications, fast and accurate solution of these equations by finite volume approach has become more important. There are two distinct paths to improve CFD algorithms: the first is to model real physics better than before by utilizing more physical terms and by decreasing our simplifying assumptions and the second is to increase the convergence rate.

This paper is related to the second approach. Explicit iteration is the simplest scheme in CFD computations, but stability requirements stall this scheme in real problems. To increase convergence rate, two different techniques are usually used. The first is using implicit methods, which have a quite wide stability limits and allow for a larger time step and therefore faster convergence. The second is using the multigrid scheme, which uses a hierarchy of grids and allows us to remove rapidly the low-frequency errors in the high-level coarse grid.

Multigrid schemes are divided to two categories: linear algebraic multigrid schemes (AMG) [1, 2], and full approximation scheme (FAS). The AMG schemes are most suitable for simple linear problems (as simulation of physical problems with linear elliptic equations) and are not recommended for nonlinear problems. In this scheme, the original problem is solved in the fine grid and the resulting errors are transferred to the coarse levels and then some operations in the coarse levels help to update the solution in the fine level. In FAS, the real nonlinear problem is solved in all levels; therefore, it needs rediscretization of the whole flow field on each coarse level. Usually this is not straightforward (especially in unstructured grids) and it still needs to improvements. Here, the FAS introduced by Brandt [1] has been used.

The first objective of this paper is to assess the possibility of combination of implicit and FAS multigrid convergence acceleration schemes, and the second objective is to introduce a rediscretization scheme for an unstructured grid. The resulting algorithm is tested in 2 and 3D applications, using first- and second-order formulations, and both cell-center and cell-vertex schemes to evaluate its advantages, applicability and robustness. Most researchers in the past have used implicit formulations in combination with AMG to solve the resulting system of equations and to show independence of convergence acceleration of both methods. The test problems used here are governed by compressible inviscid flow equations that are nonlinear convection dominant. Upwind schemes have found general popularity for these equations during recent years. Therefore, upwind FDS [3] is used here. Full-matrix implicit and FAS multigrid scheme are formulated for this problem.

Several heuristic schemes have been proposed [4–11] for generation of coarse grids, particularly in unstructured meshes. The new rediscretization scheme introduced here is applicable in both cell-center and cell-vertex unstructured grids, and is shown to be simple and appropriate. A few other techniques are also implemented so that the resulting algorithm will decrease the time complexity and provide better convergence acceleration.

2. GOVERNING EQUATIONS

The Euler equations for an unsteady compressible flow for a control volume can be expressed in the integral form as

$$\frac{\partial}{\partial t}\int_{V(t)} \mathbf{U}\,\mathrm{d}V+\int_{S(t)} \mathbf{F}_i\cdot n_i\,\mathrm{d}S=0 \tag{1}$$

where $V(t)$ is the volume of the control volume, $S(t)$ is its boundary and $\mathbf{n}$ is the unit outward vector normal to the boundary. Vector of the conserved variables $\mathbf{U}$ and the inviscid flux vector $\mathbf{F}$ are defined by

$$\mathbf{U}=\begin{bmatrix}\rho\\ \rho u_1\\ \rho u_2\\ \rho u_3\\ \rho e\end{bmatrix},\quad \mathbf{F}_i=\begin{bmatrix}\rho u_i\\ \rho u_1 u_i+p\delta_{1i}\\ \rho u_2 u_i+p\delta_{2i}\\ \rho u_3 u_i+p\delta_{3i}\\ u_i(\rho e+p)\end{bmatrix} \tag{2}$$

where the index notation has been used and ρ, p and e denote the density, pressure and specific total energy of the fluid, respectively, and u_i is the velocity of the flow in the coordinate direction x_i. The set of equations is completed by the addition of the equation of state

$$p=(\gamma-1)\rho(e-\tfrac{1}{2}u_j u_j) \tag{3}$$

which is valid for a perfect gas, where γ is the ratio of the specific heats.

3. DESCRIPTION OF THE NUMERICAL METHOD

One of the differences among the various finite volume formulations known in the literature is the arrangement of the control volume for the flow variables. The schemes used are the cell-centered and cell-vertex formulations. Both discretizations are employed in this work.

The most suitable data structure for each scheme is different. Here a face-based data structure is used, which saves volume and the conserved variables for each cell. For each face, its area vector and neighboring cell addresses are also saved. This helps us to compute only once the flux through each face.

This data structure has been constructed for finite volume schemes and is not dependent on the nodes coordinates or number of nodes of each cell or face. Such a data structure is well suited for the multigrid schemes, since the number of faces of each cell in our higher-level grids is very unusually different.

4. THE MULTIGRID SCHEME

Most relaxation schemes damp relatively quickly the high-frequency error components, but are generally slow to damp low-frequency errors. By interpolating the solution to a coarser grid, these low-frequency errors appear as higher frequencies and may be damped well by the relaxation scheme. The coarse grid can be used to compute a correction to the fine-grid solution to eliminate its low-frequency errors. By recursively using successive coarser grids, the hierarchy of low-frequency errors can be eliminated. This may result in optimal time complexity in linear cases, and to a very high acceleration for nonlinear cases.

Let us call the exact solution of Equation (1) $\tilde{\mathbf{U}}$ and the numerical solution $\mathbf{U}$ and the second term of Equation (1) $\mathbf{R}$, so the error is defined by

$$\mathbf{E}=\tilde{\mathbf{U}}-\mathbf{U} \tag{4}$$

One may write Equation (1) on the fine mesh (specified with subscript f) in discretized form as:

$$\frac{\Delta\tilde{\mathbf{U}}_{\mathrm{f}}}{\Delta t}+\mathbf{R}(\tilde{\mathbf{U}}_{\mathrm{f}})/V=0$$

We can write this equation as

$$L(\tilde{\mathbf{U}}_{\mathrm{f}})=\mathbf{f}_{\mathrm{f}} \tag{5}$$

where operator L defines the differential governing equation, while $\mathbf{f}$ is a source term. By substituting of $\mathbf{U}$ from (4) instead of $\tilde{\mathbf{U}}$ in (5) and subtracting $L(\mathbf{U})$ from both sides, one obtains:

$$L(\mathbf{U}_f+\mathbf{E}_{\mathrm{f}})-L(\mathbf{U}_{\mathrm{f}})=\mathbf{f}_{\mathrm{f}}-L(\mathbf{U}_{\mathrm{f}}) \tag{6}$$

Using operator $I_{\mathrm{f}}^{\mathrm{c}}$, all conservative variables are restricted from the fine mesh to the coarse mesh

$$L(I_{\mathrm{f}}^{\mathrm{c}}\mathbf{U}_{\mathrm{f}}+\mathbf{E}_{\mathrm{c}})-L(I_{\mathrm{f}}^{\mathrm{c}}\mathbf{U}_{\mathrm{f}})=I_{\mathrm{f}}^{\mathrm{c}}(\mathbf{f}_{\mathrm{f}}-L(\mathbf{U}_{\mathrm{f}})) \tag{7}$$

and rewriting this equation for the coarse mesh, one obtains

$$L(\mathbf{U}_{\mathrm{c}})=\mathbf{f}_{\mathrm{c}} \tag{8}$$

where

$$\mathbf{U}_{\mathrm{c}}=I_{\mathrm{f}}^{\mathrm{c}}\mathbf{U}_{\mathrm{f}}+\mathbf{E}_{\mathrm{c}} \tag{9}$$

$$\mathbf{f}_{\mathrm{c}}=I_{\mathrm{f}}^{\mathrm{c}}(\mathbf{f}_{\mathrm{f}}-L(\mathbf{U}_{\mathrm{f}}))+L(I_{\mathrm{f}}^{c}\mathbf{U}_{\mathrm{f}}) \tag{10}$$

Note that for the finest mesh $\mathbf{f}$ is equal to zero.

Several relaxations lead to a $\mathbf{U}_{\mathrm{f}}$ on the fine mesh, so one may calculate $\mathbf{f}_{\mathrm{c}}$ by Equation (10), and use it on the coarse mesh. The initial value $\mathbf{U}_{\mathrm{c}}$ in each grid level is found by restriction of the solution of (5) on the first mesh to the coarser one.

The solution on the coarse mesh, which does not include low-frequency errors, is interpolated to the finer level. $\mathbf{E}_{\mathrm{c}}$ is found by using Equation (4), and its interpolation to the finer mesh level by Equation (11) results in:

$$\tilde{\mathbf{U}}=\mathbf{U}_{\mathrm{f}}+\mathbf{E}_{\mathrm{f}},\quad \mathbf{E}_{\mathrm{f}}\approx I_{\mathrm{c}}^{\mathrm{f}}\mathbf{E}_{\mathrm{c}}=I_{\mathrm{c}}^{\mathrm{f}}(\mathbf{U}_{\mathrm{c}}-I_{\mathrm{f}}^{c}\mathbf{U}_{\mathrm{f}}) \tag{11}$$

The solution on the finer level, modified by this interpolated value, is probably relaxed a few times, and the whole process is repeated respect to the next fine level, till we reach to the last level. Both restriction and prolongation operators need to be conservative, otherwise the transferred values may include some sink or source terms, which may delay convergence.

5. THE IMPLICIT SCHEME

The governing equation (1) is discretized using a finite volume formulation. The inviscid fluxes at faces are computed using the Roe scheme. This finite volume approximation yields in (5), which is a semi-discrete system of nonlinear equations. The main function here is the solution of Equation (1), which is rewritten by (5) and (8) on different grid levels. Since this equation is nonlinear, we should use an iterative scheme to solve it. This iterative scheme (which describes our relaxation scheme) could be either explicit or implicit. Here, the implicit scheme is used to achieve higher acceleration. A family of implicit schemes for Equation (5) is

$$\frac{\mathbf{\Delta U}}{\Delta t}+\mathbf{R}^{n+1}/V=0, \quad \mathbf{\Delta U}=\mathbf{U}^{n+1}-\mathbf{U}^{n} \tag{12}$$

Equation (12) represents a system of nonlinear coupled equations to achieve the steady-state solution, which can be linearized in time by setting

$$\mathbf{R}^{n+1}=\mathbf{R}^{n}+\left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|^{n}\mathbf{\Delta U} \tag{13}$$

This yields the following system of linear equations:

$$\left[\frac{V}{\Delta t}\mathbf{I}+\left.\frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right|^{n}\right]\mathbf{\Delta U}=-\mathbf{R}^{n} \tag{14}$$

The Jacobian of **R** in the left-hand side may be estimated by an approximate inviscid first-order scheme like Van Leer's to accelerate the convergence. Only on the finest grid, the value of **R** in the right-hand side could be approximated by a second-order and high-resolution computation to keep the desired level of accuracy [12]. The resulting system of linear equations is solved by the SGS method.

6. THE AGGLOMERATION TECHNIQUES

One of the main challenges for the multigrid schemes is the coarsening procedure, especially on unstructured grids. Most schemes are more suitable for node-centered methods, which are not used here. For structured grids, the obvious option is to double the grid size in each direction, but for unstructured mesh it is not obvious which coarsening procedure provides the highest acceleration. The grid shape is also very important and bad grid quality may degrade the convergence.

Agglomeration techniques have received markable notice in recent years [4–7]. One of the agglomeration techniques is based on node deletion [8]. This method is generally simple, but computation of the conservative transfer function is more expensive. There are other agglomeration schemes [5], which are based on deletion of nodes in the lower-level grid and retriangulation of the remaining mesh. In these algorithms the coarsening ratio could be increased easily, but the resulting interpolations could be very expensive. This time complexity is more prohibiting in moving or 3D applications, and also preservation of the boundary geometry is more difficult.

The other techniques use a merging scheme, so that fine mesh cells are mixed and agglomerated to produce new coarse mesh. Here, the transfer functions are easily computed. Smith [9] used this technique in their multigrid implementation and similar works by Lallemand *et al.* [10] and Mavriplis and Venkatakrishnan [11] and many others have been inspired by this technique. In the present work, an agglomeration technique was implemented which has been designed for cell-centered control volumes and has been inspired also from merging scheme. The procedure could be easily extended to 3D. A trick was also introduced for 2D geometries that results in better quality in shape of coarse grids.

7. A FAST AND SIMPLE AGGLOMERATION TECHNIQUE

The algorithm presented here starts with an unstructured triangular mesh. In the cell database each cell has a flag, which shows whether that cell is selected or unselected. Each node in the node database has a similar flag. Initially all flags are set to 'unselected'. The algorithm of this method follows:

(1) Select a starting point in the mesh. A good starting node could be a node with the highest number of common elements. Add this node to the list of candidate nodes (which is abbreviated by **CN**).
(2) While the list of candidate nodes is not empty do steps 3 to 4.
(3) For each candidate node, construct a new coarse element by:

    (3.1) Find all elements including the candidate node.
    (3.2) If these elements are unselected, put them in the list of 'surrounding elements' (**SE**).
    (3.3) If **SE** has one or less member, remove the candidate node from **CN**. Go to step 2.
    (3.4) Change flag of all members of **SE** to 'selected'. (This will remind us in the coarse mesh in the future that this cell of fine mesh is already taken care of.)
    (3.5) Merge all members of **SE** in a new cell. Then add this new cell to the list of cells of the next level grid.

(4) Find and prioritize the new list of candidate nodes and add to the end of **CN**.

    (4.1) Find the list of fine mesh nodes (called **FN**) which are directly connected to the new coarse mesh cell (called **NCMC**) which are 'unselected'.
    (4.2) For each member of **FN**, find the corresponding coarse cell (not generated yet).
    (4.3) Find the number of common edges between coarse cells of members of **FN** and the **NCMC**. (Those that have no common edge, should be removed from **FN**.)
    (4.4) Sort list of **FN**, based on this number (descending).
    (4.5) Add this list to the end of **CN**. Based on our experiences this algorithm, based on local optimization, is the best among possible alternative algorithms.

(5) Obviously, there remain some cells that have not contributed to the coarsening procedure. This final process will remove these 'unselected' spaces:

    (5.1) Sort all resulting new coarse cells based on (ascending) number of their subelements.
    (5.2) For each member of this sorted list.

(5.2.1) Add all neighbors that are ‘unselected’ fine mesh elements.
(5.2.2) Reconstruct the coarse grid database using these new subelements.

(5.3) Continue step 5.2, until no ‘unselected’ element exists.

In a 3D case, Step 5 could be simplified, to make it faster. While ‘unselected’ cells exist one may check all of its neighbors and add it to one of them which has the minimum subelement count.

To clarify the algorithm, the pseudocode of this algorithm is shown here.

Note that this scheme reduces to the optimum agglomeration scheme for a structured mesh, i.e. by this algorithm a 64×64 structured grid will reduce to 32×32 and then to 16×16. In this special case, the coarsening ratio is 4 for structured 2D mesh; but for unstructured mesh, depending to initial grid distribution and quality, it is about 6 in the first coarsening level and about 3 for the next levels. There are many other alternatives with different coarsening ratios, but finding the best coarsening ratio requires further investigation.

8. A GRID PRECONDITIONER FOR 2D TRIANGULAR MESHES

A special alternative is investigated here. Since for a structured mesh, as explained above, the coarsening scheme is somehow optimal, a special algorithm is developed that produces quadrilateral elements from each pair of triangles. These quadrilateral elements are used as the initial grid for the previous algorithm. The algorithm of this preconditioner procedure is:

(1) Select an initial element in the base fine mesh. It should be a 2D triangular unstructured mesh. Since an appropriate grid for external flows usually has a regular distribution of elements close to the far-field boundary, it is better to start with a cell on the far-field boundary.
(2) As mentioned earlier, each cell has a ‘selection’ flag, which is initially set to ‘unselected’.
(3) Define the best neighbor as a neighbor of the corresponding cell, which is ‘unselected’ and it has the least number of ‘unselected’ neighbors (zero is not acceptable). If there are more than one, select only one of them. If the best neighbor is not found, go to step 6.
(4) Set flag of the best neighbor to ‘selected’ and merge it in the original cell as a new quadrilateral element.
(5) Select the best neighbor as the new original cell and go to step 3.
(6) Here, there is no best neighbor. Through the list of added quadrilaterals:

(6.1) Select the last quadrilateral.
(6.2) If the best neighbor of the quadrilateral is found and has a best neighbor too, select that best neighbor as the new original cell and go to step 3.
(6.3) Select the next element from the end of quadrilaterals list and go to 6.2.

(7) Remove all ‘unselected’ triangles from domain by the removal procedure described in Section 7, Step 5. With this procedure some elements with five or more edges may be produced.

When this preconditioner is applied, a new grid with a coarsening ratio about 2 is constructed. If this grid is used as the initial fine grid, the resulting coarsening ratio for the first coarse grid will be about 8, while for the next levels it will be about 3 (Figures 2–4).

Moreover, the shape of the agglomerated grid with this preconditioner has less saw-tooth boundaries, which sounds to be more efficient.

```
G is current Grid
CG is Coarse Grid
CN is List of candidate nodes
/* the starting node has maximum count of
   elements which include it */
C=Max(G.Nodes,NeighborElements().Count())
CN.Add(C)
Foreach C in CN do
   CE=CreatCoarseElement(C)
   If CE.SE.Count() <=1 then
    //.SE is subelements list
   CN.Remove(C)
   Else
   CG.Add(CE)
     Foreach E in CE.SE do
     E.Selected=true
     E.HCE=CE
     //.HCG is coarse element of the
element
    End
  End
End

FN is a List of Nodes

Foreach N in C.NeighborNodes() do
     FN.Add(N.NeighborNodes()
         ,Not Duplicate
         ,Not in CN List
         )
End

Sort(Descending,FN,GetCommonFaceCount,CE)
CN.Add(FN)
End
/*Find a coarse element to
add these remaining elements */
Sort(Acsending,CG,SE.Count)

Foreach CE in CG do
   Foreach UE in CE.NeighborElements() do
      If not UE.Selected then
      CE.SE.Add(UE)
      UE.Selected=true
      End
   End
End

End
Stop .
```

```
Function GetCommonFaceCount(N,CE)
Count=0
NCE= CreatCoarseElement(N)
    Foreach E in CE.SE do
    If E.IsNeighborOf(NCE) then
    Count=Count+1
    End
Return Count
End


Function CreatCoarseElement(C )

CE is a CoarseEleman
CE.SE is the List of SubElements


Foreach E in C.NeighborElements()do
    If not E.Selected then
      CE.SE.Add(E)
     End
   End

     If CE.SE.Count()<=1 then
       Return
     End

     Foreach E in CE.SE do
          E.Seletecd=true
     End
     CE.MergeSubElements()
 Return CE
 End

 Function FastRemoveUnSelected()
 HasUnSelected=true
 While HasUnSelectd do
 HasUnSelected=false
  Foreach E in G do
    If not E.Selected then
        Min=1e10
   Foreach EN in ER.NeighborElements
  ()do
        If EN.HCE.SE.Count() < Min then
         Min= EN.HCE.SE.Count()
         CG=EN.HCE
       End
  End

  If Min >0 and not 1e10 then
    CG.SE.Add(ER)
  Else
        HasUnSelected=true
  End
 End
 End
End
```

9. EXTENSION TO CELL-VERTEX GRID AND GENERALIZATION TO THREE DIMENSIONS

The scheme presented in Section 7 could be generalized to 3D grids. Deleting expensive operations of Step 5.1, one may improve its performance over large grids. The grid preconditioner scheme (presented in Section 8) has the advantage of producing quadrilateral cells as our initial grid set. The agglomeration of this grid will produce a suitable coarse grid. This grid is similar to the structured grid used in external flow solvers, and the boundary is smoother, and the saw-tooth boundaries are less generated. The advantages of this are more explained in Section 10. Since a combination of two tetrahedral cells does not necessarily produce automatically a hexahedron cell, therefore, there is no guarantee for the production of an appropriate cell. 3D extension is not trivial and will not produce satisfactory results.

The agglomeration scheme, presented for 2D cell-centered case, could be generalized to 2D or 3D cell-vertex grids. In cell-centered scheme, we used elements around an arbitrary vertex of a cell, i.e. we integrate elements that have a node in common. In cell-vertex schemes, each cell is initially produced around each vertex, and cells that have a centroid of an element in common are integrated to produce a larger cell. Since here operations explained in Section 7.1 is not required (all elements include three cells (four cells, in 3D), except a few boundary cells), one may start the operation from an internal element. The other operations are similar to cell-center scheme, if one uses cells around element centroid points, instead of cells around nodes. Examples of these extensions to 3D cases are reported in result section.

10. THE MULTIGRID IMPLEMENTATION

After the agglomeration process, the multigrid solution is started. The most important parts of this process are the method of computation, transformation and saving of the source term **f** among different levels. Value of R/V (the second term in Equation (1)) and **f** should be stored separately for each cell. Conservative transformation operators would result in faster convergence. This is performed using area of each cell, as:

$$\mathbf{U}_{\mathrm{c}} = \frac{\sum U_i^{\mathrm{f}} V_i^{\mathrm{f}}}{\sum V_i^{\mathrm{f}}} \tag{15}$$

where V_i^{f} is the contribution of the volume of the ith cell of the fine mesh in the coarse mesh. In agglomeration scheme, since only a few complete fine cells make up a coarse cell, the above procedure could be achieved very easily, and the process is very fast.

Many tricks may be used to decrease the computation costs or to increase the acceleration rate. Most of the computations are devoted to the flux computations, and there are many ways to do it [3, 13]. One may observe that the solution in the coarser meshes is to remove the low-frequency error modes; therefore, fast low-order methods are quite acceptable on coarse meshes. We have used the VanLeer FVS method [14] for our flux computations on second- or higher-level grids. For lower computational and saving costs, matrix-free methods could be used to solve the equations made in our implicit method [12]. The other technique, which saves us lots of flux computation, is based on features of agglomeration schemes. In these methods, especially in higher-level grids, each two neighboring cells may share in several edges, which simply results in lots of flux computations.

To decrease these computations, an equivalent edge is defined everywhere, except on the boundary faces, to keep accuracy of the original grid geometrical modeling, which is found by a combination of all common edges.

$$\sum \mathrm{F}_i^{\mathrm{LR}} \cdot \mathrm{dA}_i^{\mathrm{LR}} \approx \vec{\mathrm{F}}^{\mathrm{LR}} \cdot \sum \mathrm{dA}_i^{\mathrm{LR}} = \vec{\mathrm{F}}^{\mathrm{LR}} \cdot \mathrm{dA}_{\mathrm{Equivalence}}^{\mathrm{LR}} \tag{16}$$

The effect of these techniques on convergence acceleration is investigated in the results section.

11. RESULTS

To show capabilities of the above scheme and to compare it with other schemes, solution of the compressible transonic flow around 2 and 3D bodies are considered [15]. To assess quality of the agglomeration scheme and its coarsening ratio, a few regular and nonregular boundary shapes are used. A few problems are also analyzed to verify the validity of the developed computational algorithms. Then, for 2D cases, the agglomeration and 2D preconditioner schemes are applied and assessed, and implicit and explicit convergence rates are compared. A few techniques, introduced in Section 10, and their effectiveness are also examined here. For extension to three dimensions, implicit and explicit algorithms are applied to solve a standard test problem. Here a cell-vertex scheme is used and compared with the original cell-center schemes.

11.1. Verification

Transonic flow around an NACA0012 airfoil is considered here. The angle of attack is 1° and $M = 0.85$. Results of this test case are reported in many references, as [15]. To investigate grid independency, the original method is first used on two unstructured grids with, respectively, 21 824 and 4731 cells, and results are compared with [16] and Fluent. Accuracy of the results was found to be satisfactory (Figure 1(a)). To investigate extension in three dimensions, transonic flow around Onera M6 wing is examined. The angle of attack is 3.06° and M is equal to 0.8395. For technical reasons, the final agglomerated grid could not be effectively shown. A result of a first-order computation for a fairly fine mesh over the wing is shown in Figure 5, and is compared with experimental results [17, 18] (Figure 1(b)). The initial grid includes 313 424 cells and 66 467 nodes.

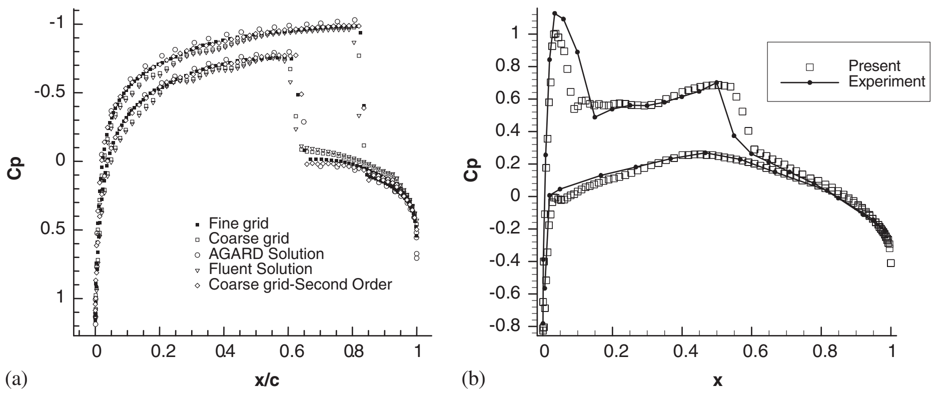

Figure 1. Comparison of pressure coefficient distribution with the existing results.

11.2. The agglomeration scheme

Figures 2, 3 and 6 show the coarsening ratio distribution for each level of multigrid method. These figures show how different agglomeration schemes affect the coarsening ratios.

To show the capabilities of this scheme in a less organized initial grid, a nonconventional boundary shape is selected and triangulated (Figure 2). The distribution of coarsening ratio is shown in Figure 2. In an ordered grid, this ratio is 6 for the first coarsening procedure, and then decreases to 3. One observes that the coarsening ratio is not significantly different (Figure 6).

It is known that these procedures affect the convergence acceleration, but no complete analysis on this subject could be found in the literature.

Two schemes are introduced in Sections 7 and 8. Here we analyze the performance of these schemes and their effects on convergence acceleration. As a criterion for convergence, we consider in all cases the time (or iteration) required to reach to 10^{-13} in the residual of computations (13 order of magnitude convergence). The residual is defined as L_2 norm of $\|\partial U/\partial t\|_2$.

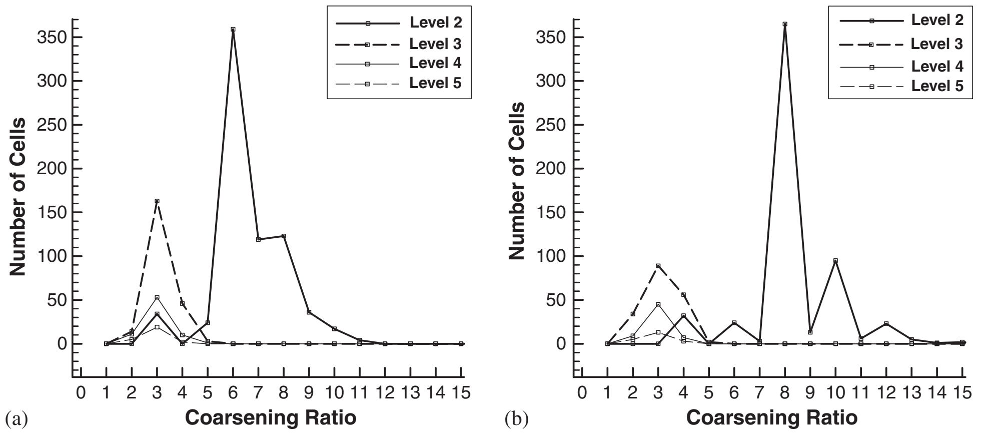

Figure 2. Distribution of coarsening ratio for 2D unstructured grid by: (a) the first agglomeration and (b) applying the grid preconditioner.

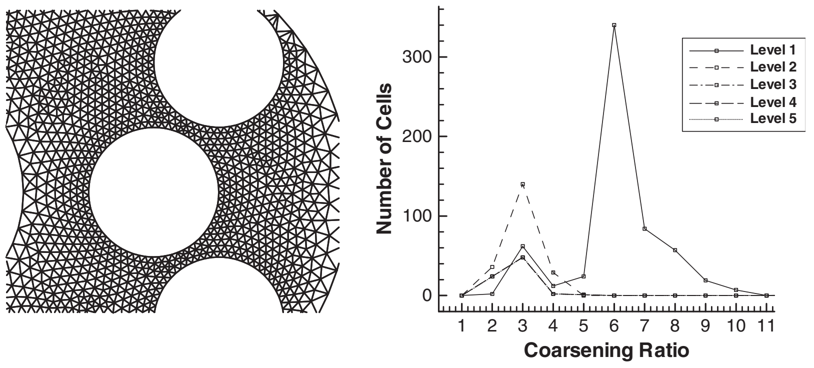

Figure 3. Application of the agglomeration scheme to a more complex geometry and its coarsening ratio.

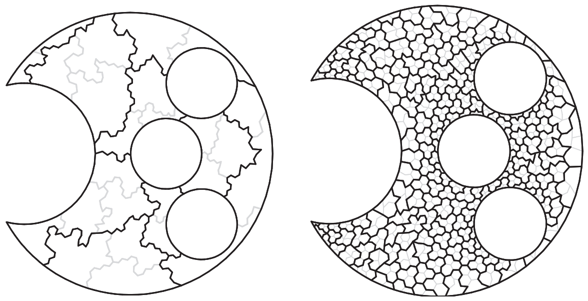
Figure 4. Coarse grid in two different agglomeration levels for a more complex geometry.

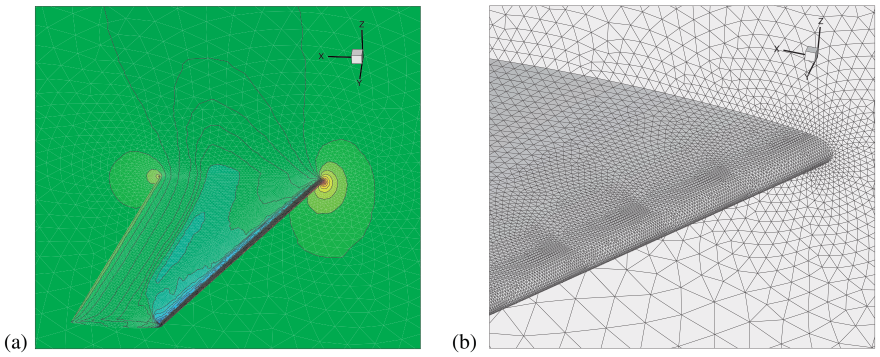

Figure 5. (a) Pressure distribution over ONERA wing and (b) part of the used grid.

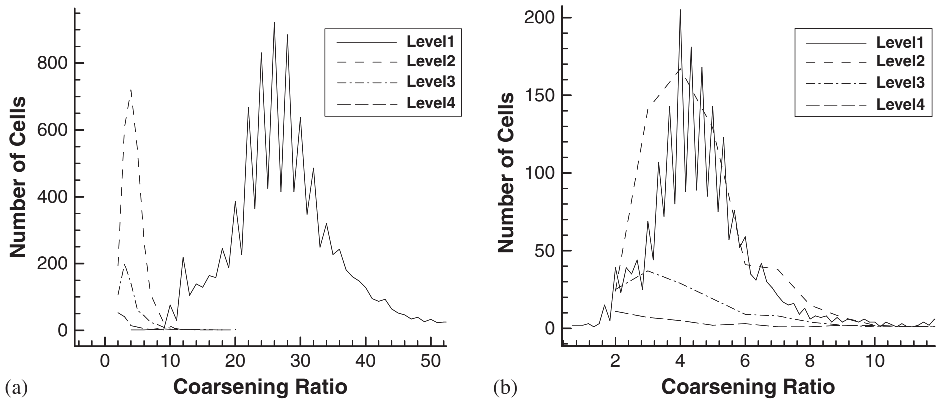

Figure 6. Distribution of coarsening ratio for 3D unstructured grid by the first agglomeration: (a) cell-center discretization and (b) cell-vertex discretization.

A V cycle is used for multigrid computations, and after each restriction, four relaxations are performed on the coarser mesh and after each prolongation, three relaxations are performed on the finer mesh.

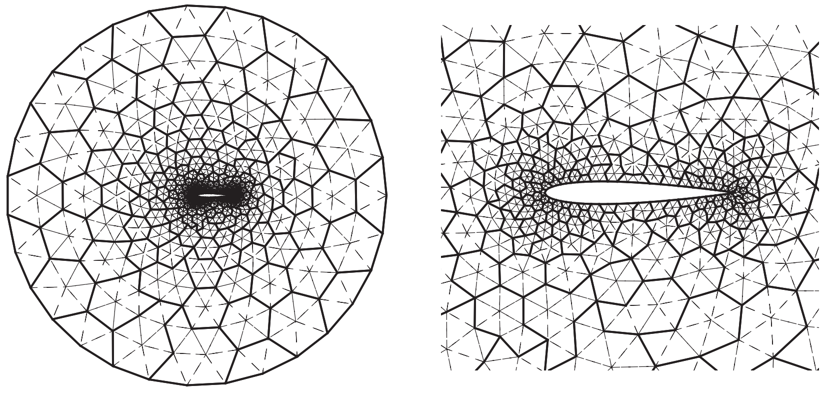

Figure 7. The first coarse grid produced by the first agglomeration scheme.

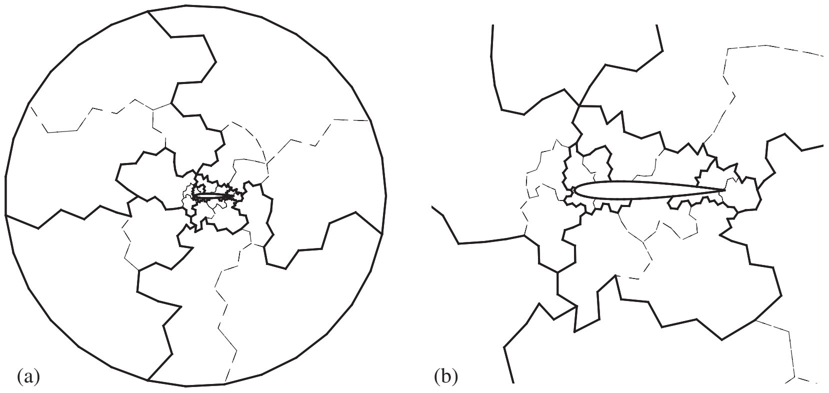

Figure 8. The highest-level coarse grid produced by the first agglomeration scheme: (a) overall view and (b) grid around the airfoil.

Figures 4, 7 and 8 show the original and the second-level grids for the simple agglomeration scheme. As explained in Section 7, the coarsening ratio is about 6 in the first coarsening, and about 3 in the next levels. Figures 9–11 show the original grid and the temporary grid produced for the grid preconditioning method.

Figure 12 schematically shows the effect on convergence rate of increasing the number of meshes in the multigrid hierarchy by applying all convergence improvements on implicit and explicit solvers. Figure 12(a) shows strong multigrid influence on explicit solvers and (b) shows influence of multigrid acceleration on implicit solver.

Since the implicit methods and multigriding accelerate convergence independently and in this context their conjunction would be analyzed, we compute all speedup with respect to explicit solution on original fine grid.

Figure 13 shows effect of first agglomeration scheme (triangle based) and preconditioned agglomeration (quadrilateral based) on the speedup and time complexity of the solution method.

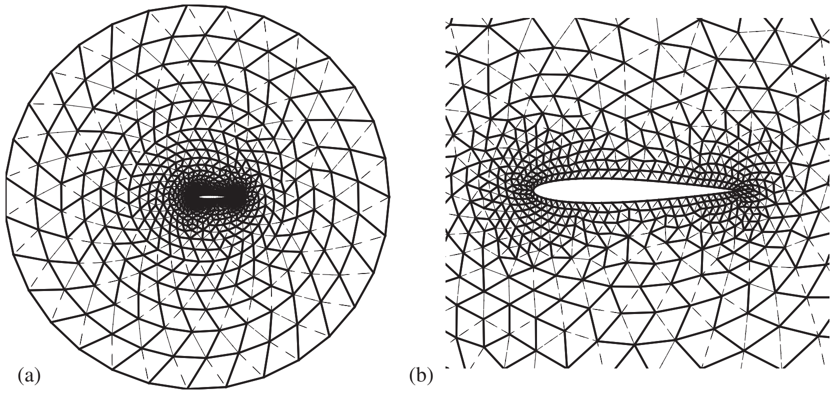

Figure 9. The preconditioned grid: (a) overall view and (b) grid around the airfoil.

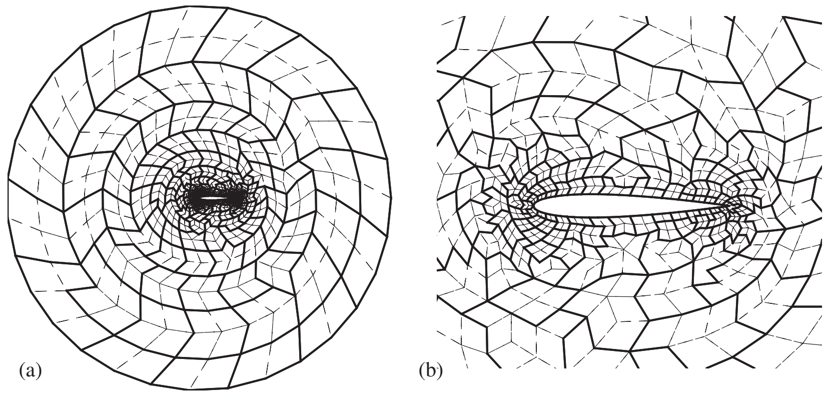

Figure 10. The first coarse grid based on the preconditioned grid: (a) overall view and (b) grid around the airfoil.

Figure 13(a) compares iteration number for these two schemes and shows that they are fairly similar. Figure 13(b) compares the results of these schemes and shows that for explicit solver 2 levels MG method, both schemes are the same, but for 3 or 4 levels MG, the second scheme (quadrilateral based) performs slightly worse and the best speedups are, respectively, 6.8 and 5.5. However for implicit solver the quadrilateral based grids have higher speedup in all levels.

11.3. The implicit solver

Here we study the effects of the implicit solver on convergence acceleration of the MG scheme, and explore effects of different agglomeration schemes on effectiveness of implicit solution. Other researchers have also studied application of implicit solvers to MG procedure (e.g. [19–22]).

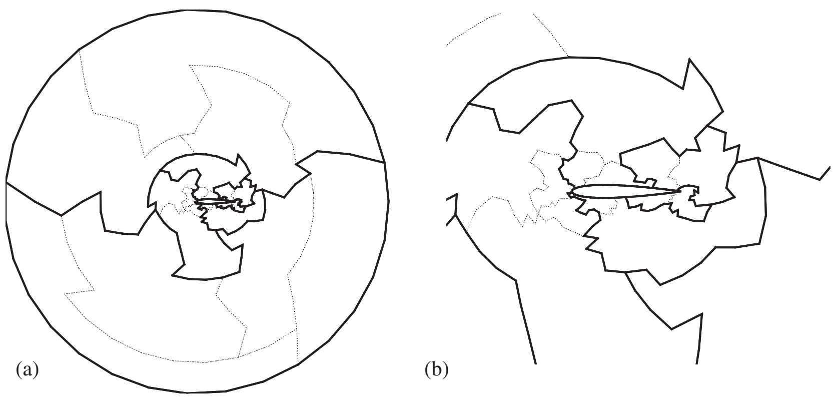

Figure 11. The third coarse grid based on the preconditioned grid: (a) overall view and (b) grid around the airfoil.

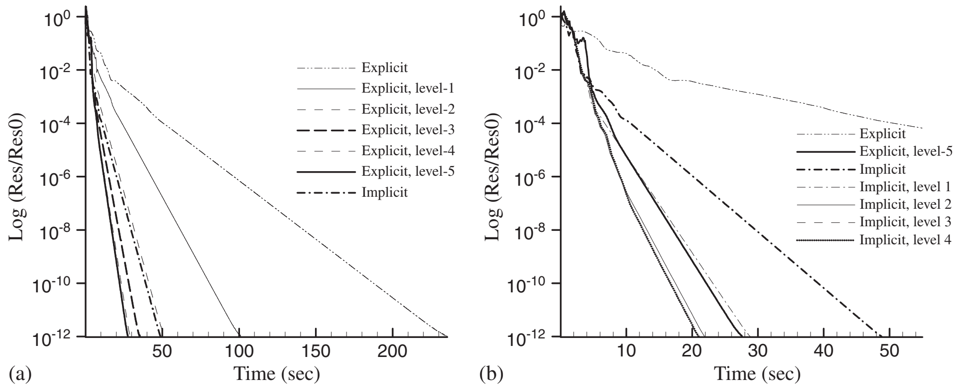

Figure 12. Convergence history of multigrid acceleration for: (a) explicit and (b) implicit solvers.

Figures 13 and 14 compare the results of implicit and explicit schemes, for both MG methods. One observes that

- The savings in the iteration number when explicit solution is used steadily increases by increasing the number of MG levels (see Figure 13(a)), but this ratio for implicit solution is limited (see Figure 13(b)).
- Figure 14(a) shows that for 3 levels triangle-based MG, the speedup resulted from implicit solution is (9.8/4.8) but for quadrilateral based it is (10.5/4.8).
- The optimum number of MG grid levels decreases when implicit solution is applied. This optimum number is the function of many parameters. In this test case, it is three for quadrilateral-based grids and four for triangle-based grids. We observed higher optimum number for low Mach number cases or finer grids.
- The total speedup for quadrilateral implicit 3 levels MG scheme is (10.5), while the best speedup for explicit solvers is observed for 5 levels MG and is equal to (6.8).

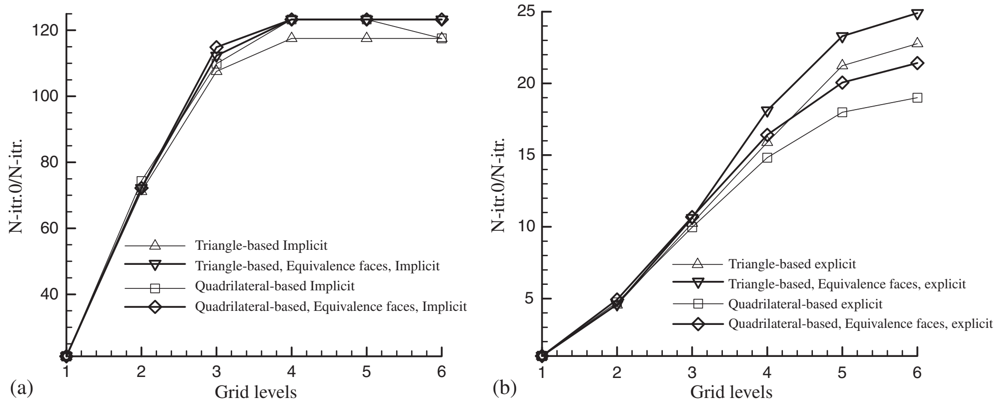

Figure 13. Comparison of total iteration number: (a) implicit multigrid and (b) explicit multigrid.

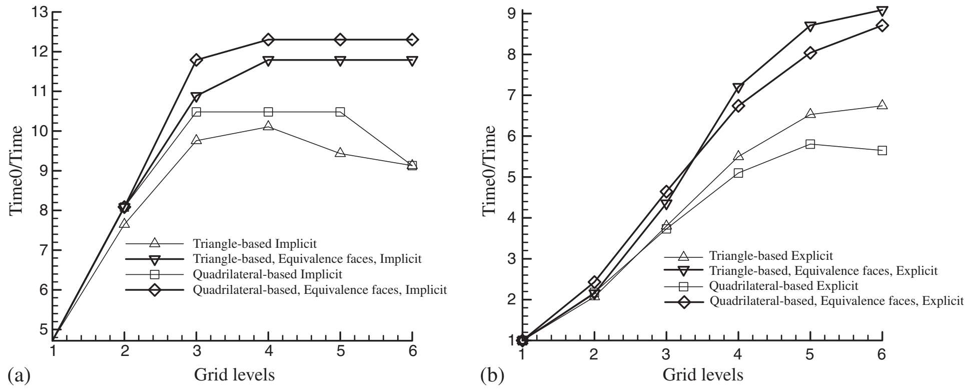

Figure 14. Speedup in CPU time: (a) implicit multigrid and (b) explicit multigrid.

11.4. Other computational techniques

Two techniques are proposed in Section 10 to increase the convergence rate of implicit multigrid schemes. Figures 13 and 14 show the effect of using the equivalent faces instead of original faces. One observes that

- For explicit solvers, this effect is easily seen in Figure 14(b), which result in a speedup increase of (9.0/6.8) for triangle-based MG and (8.7/5.8) for quadrilateral-based.
- The effect of equivalent faces on total iteration number is negligible (as we expect no change) (Figure 13).

11.5. Second-order computation

To have a fair comparison of time speedups of high-order explicit solutions, the solution is advanced in time using a multi-stage Runge–Kutta scheme, which has good stability and damping properties.

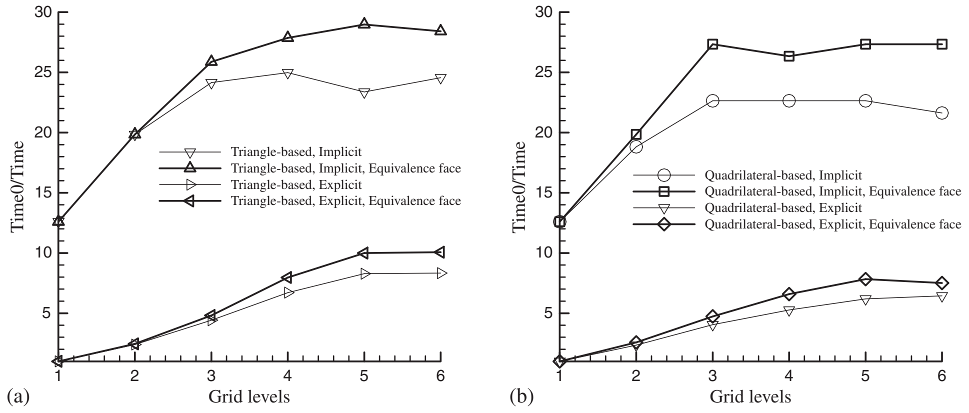

Figure 15. Comparison of speed up in CPU time for different agglomerations for second-order computations: (a) triangle based and (b) quadrilateral based.

To achieve higher convergence acceleration, the second-order calculation is only performed in the original fine grid and computations over the coarse agglomerated grids remain first order. A TVD scheme, using a multidimensional linear reconstruction approach [23], is used to achieve high order of accuracy. The Minmod limiter, based on conservative variables, is used to suppress the spurious oscillations in the vicinity of discontinuities. For the above standard transonic flow, Figure 15 shows the obtained speedups on the same grid. The implicit method has produced very high speedups for all MG levels, and using the equivalent face technique has effectively increased the speed for both explicit and implicit methods.

11.6. 3D extensions

The agglomeration scheme is first generalized to three dimensions. In the cell-center case, cells connected to a node are integrated to produce a coarser cell. This will result in very high coarsening ratio up to 30, for the first level, and regular coarsening ratios of around 5 for the next levels (Figure 6(a)). In the cell-vertex case, since we integrate control volumes around vertices of each tetrahedral, the coarsening ratio for the first level is around four, but it has a distribution that is shown in Figure 6(b). This ratio is similar to the cell-center case for higher levels.

Figure 16 shows the convergence acceleration provided by this multigrid scheme, based on CPU time. One observes a speedup of more than four times for two levels multigrid. The main reason for this is the high coarsening ratio for first level of coarsening. For cell-vertex method, this acceleration ratio for two levels multigrid is only about two. For explicit four levels multigrid, cell-vertex schemes has an acceleration ratio of about 3.5, while cell-center schemes are accelerated about five times. For implicit computations, similar trends are observed.

12. CONCLUSION

A new agglomeration algorithm to be used in multigrid with FAS is introduced. These algorithms produce well-shaped grids for all levels of coarse meshes. This simple scheme is applied in

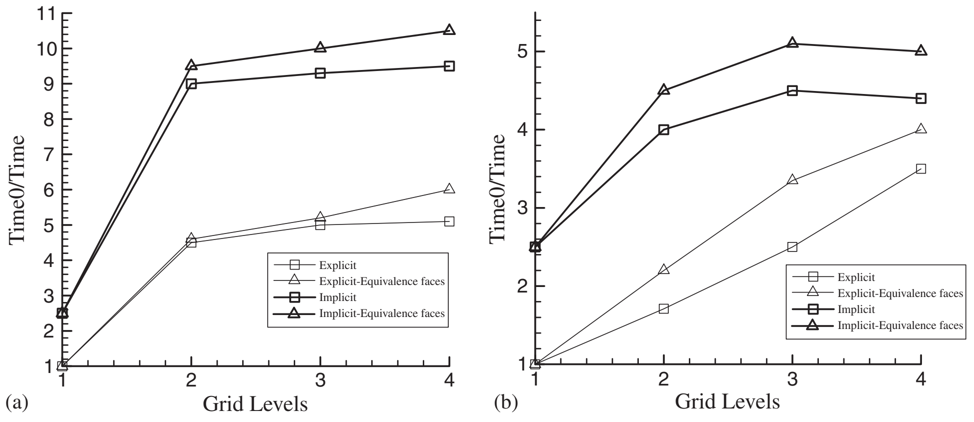

Figure 16. Comparison of speed up in CPU time for flow simulation over the ONERA M6 grid: (a) cell-center and (b) cell-vertex.

2D/3D unstructured grids with cell-center and cell-vertex discretizations. The generated grid is verified to be appropriate for acceleration of implicit solution of the inviscid flow equations over them. It is shown that convergence acceleration mechanism due to the implicit formulation is independently effective. Appropriate convergence acceleration is observed in the application of higher-order formulation in 3D and cell-vertex computations. Some other techniques such as edge/face collapsing and using fast low-order computations in coarse levels are also applied and their advantages are shown.

ACKNOWLEDGEMENTS

This research is partly supported by the research office of Sharif University of Technology.

REFERENCES

1. Brandt A. *Multigrid Methods*. Lecture Notes in Mathematics. Springer: Berlin, 1982.
2. Briggs WL. *A Multigrid Tutorial*. SIAM: Philadelphia, 1987.
3. Roe PL. Approximate Riemann solvers. *Journal of Computational Physics* 1981; **43**:357–372.
4. Venkatakrishnan V, Mavriplis DJ. Implicit solvers for unstructured meshes. *AIAA 91-1537-CP*, June 1991.
5. Guillard H. Node nested multigrid with Delaunay coarsening. *INRIA Report No. 1898*, 1993.
6. Mavriplis DJ, Venkatakrishnan V. Agglomeration multigrid for two-dimensional viscous flows. *Journal of Computational Physics* 1995; **24**:553–570.
7. Mavriplis DJ, Jameson A, Martinelli L. Multigrid solution of the Navier–Stokes equations on triangular meshes. *NASA CR-181786*, February 1989.
8. Gooch CFO. Multigrid acceleration of an upwind Euler solver on unstructured meshes. *AIAA 95-1452*, vol. 33(10), 1995; 1822–1827.
9. Smith WA. Multigrid solution of transonic flow on unstructured grids. In *Recent Advances and Applications in Computational Fluid Dynamics*, Baysal O (ed.). *Proceedings of the ASME Winter Annual Meeting*, November 1990. ASME: New York, 1990.
10. Lallemand M, Steve H, Dervieux A. Unstructured multigridding by volume agglomeration: current status. *Computers and Fluids* 1992; **21**:397–433.
11. Mavriplis DJ, Venkatakrishnan V. A 3D agglomeration multigrid solver for the Reynolds-averaged Navier–Stokes equations on unstructured meshes. *AIAA Paper 95-0345*, 1995.

12. Luo H, Baum JD, Louhner R. An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grid. *Computers and Fluids* 2001; **30**:137–159.
13. Anderson WK, James L, Whitfield DL. Three-dimensional multigrid algorithms for the flux-split Euler equations. *NASA TP 2829*, November 1988.
14. Van Leer B. Flux vector splitting for the Euler equations. *Proceedings of the 8th International Conference on Numerical Methods in Fluid Dynamics*. Springer: Berlin, 1982.
15. Bortoli ALD. Multigrid based aerodynamical simulations for the NACA0012 airfoil. *Applied Numerical Mathematics* 2002; **40**:337–349.
16. Advisory Group for Aerospace Research and Development (AGARD), Test Cases for Inviscid Flow Field Methods. *AGARD Advisory Report No. 211*, 1985.
17. Schmitt V, Charpin F. Pressure distributions on the ONERA-M6-wing at transonic mach numbers. Experimental data base for computer program assessment. *Report of the Fluid Dynamics Panel Working Group 04, AGARD AR 138*, May 1979.
18. Carre G. An implicit multigrid method by agglomeration applied to turbulent flows. *Computers and Fluids* 1997; **26**:299–320.
19. Caughey D. Implicit multigrid Euler solutions with symmetric total-variation-diminishing dissipation. *AIAA 93-3358-CP*, 1993.
20. Swanson RC, Turkel E, Rossow C-C. Convergence acceleration of Runge–Kutta schemes for solving the Navier–Stokes equations. *Journal of Computational Physics* 2007; **224**(1):365–388.
21. Amaladas JR, Kamath H. Implicit and multigrid procedures for steady-state computations with upwind algorithms. *Computers and Fluids* 1999; **28**(2):187–212.
22. Liang C, Kannan R, Wang ZJ. A p-multigrid spectral difference method with explicit and implicit smoothers on unstructured triangular grids. *Computers and Fluids* 2008; in press.
23. Darwish MS, Moukalled F. TVD schemes for unstructured grids. *International Journal of Heat and Mass Transfer* 2003; **46**:599–611.