INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 22, 411–428 (1996)

THE PARALLEL BLOCK ADAPTIVE MULTIGRID METHOD FOR THE IMPLICIT SOLUTION OF THE EULER EQUATIONS

NIKOS G. PANTELELIS AND ANDREAS E. KANARACHOS

National Technical University of Athens, PO Box 64078, 15710 Athens, Greece

SUMMARY

A method capable of solving very fast and robust complex non-linear systems of equations is presented. The block adaptive multigrid (BAM) method combines mesh adaptive techniques with multigrid and domain decomposition methods. The overall method is based on the FAS multigrid, but instead of using global grids, locally enriched subgrids are also employed in regions where excessive solution errors are encountered. The final mesh is a composite grid with uniform rectangular subgrids of various mesh densities. The regions where finer grid resolution is necessary are detected using an estimation of the solution error by comparing solutions between grid levels. Furthermore, an alternative domain decomposition strategy has been developed to take advantage of parallel computing machines. The proposed method has been applied to an implicit upwind Euler code (EuFlex) for the solution of complex transonic flows around aerofoils. The efficiency and robustness of the BAM method are demonstrated for two popular inviscid test cases. Up to 19-fold acceleration with respect to the single-grid solution has been achieved, but a further twofold speed-up is possible on four-processor parallel computers.

KEY WORDS: composite grids; adaptive grids; multigrid; parallelization; Euler; implicit scheme

1. INTRODUCTION

Although multigrid methods were introduced as grid adaptation techniques, they have been established only as fast and efficient solvers for large-scale computations. So far only a few mesh adaptive multigrid schemes have been proposed, e.g. the multilevel adaptive techniqaue (MLAT),[1] the fast adaptive composite (FAC) grid[2,3] and a few others,[4–7] but their domain of application is restricted to elliptic-type equations. Regarding the development of adaptive schemes for hyperbolic systems of equations, few attempts have been made to take advantage of the favourable multigrid concept for the acceleration of the solution.[8] On the other hand, great advantages have been pointed out for the use of truncation error prediction as a reliable error sensor for mesh adaptation procedures, although few studies have presented numerical results[9–11] and few theoretical analyses exist.[12,13]

The mesh adaptation methods are classified as mesh-moving and mesh-embedding methods, but a combination of them is also possible. The mesh-moving approach provides the best solution for a given number of points, whereas the mesh-embedding technique aims to attain the prescribed level of accuracy for the least computational cost. The structured grid-embedding technique employs only uniform subgrids,[4,6,9,10] whereas the semi-structured grid-embedding technique constructs a fully irregular mesh of quadrilaterals with refinements exactly where needed.[5,8] Finally, the completely unstructured mesh refinement technique utilizes only triangular volumes as in the unstructured finite element method. Each of the above grid adaptation methods has its own advantages ranging from the ability to handle complex domains to the ease of implementation and fast convergence.

CCC 0271–2091/96/050411–18

Received December 1994
Revised May 1995

For the present method the implementation of the mesh-embedding technique using rectangular blocks has been adopted. This technique has been used in combination with multigrid methods only for elliptic equations successfully.[4–7] It has been proposed without multigrid for the solution of hyperbolic-type equations of inviscid flows[9] and hydrodynamics.[10] In these studies,[9,10] rectangular patches of higher grid resolution are introduced in regions with increased truncation error levels. However, multigrid methods should always be considered for the development of mesh adaptive methods, because the existence of several grid scales improves the single-grid solution convergence rates significantly. Furthermore, regarding the composite grid solution techniques in hyperbolic equations,[9,10,14] the explicit solution schemes are dominant, since implicit methods require even more complicated schemes.

In the present study a dynamic mesh adaptive method, the block adaptive multigrid (BAM) method,[11] is presented incorporating a reliable error prediction device and a composite grid–multigrid solver. The method is based on the full multigrid (FMG) scheme. Starting from an acceptable coarse mesh, the solution creates subgrids of finer grid levels only where required. In this way an adapted non-uniform grid is decomposed to uniform subgrids where common solvers can be used. The composite grid structure is handled entirely by the multigrid method, whereas for the parallelization of the code an alternative domain decomposition is used for the achievement of load balance. For the integration and relaxation of the time-marching Euler equations an unfactored implicit upwind finite volume scheme has been used.[19] The proposed BAM method is verified for two complex transonic inviscid cases. Using the proposed method, robust and accurate solutions can be obtained in a reduced number of work units (18-fold acceleration with 4·5 times fewer volumes with respect to the single-grid calculations).

The rest of the paper is organized as follows. In the next section the Euler equations and the finite volume discretization method are analysed. The block adaptive multigrid (BAM) method is composed of three main parts: the non-linear multigrid solver is presented in Section 3, the truncation error sensor for the prediction of the solution error is discussed in Section 4 and the composite grid solver is analysed in Section 5. In Section 6 an alternative domain decomposition method is proposed to take advantage of multiple processors in situations where mesh adaptive techniques have faced significant problems. Finally, results are presented in Section 7 and conclusions are drawn in Section 8.

2. GOVERNING EQUATIONS

The general inviscid flow is described by the Euler equations, which can be solved using the very popular time-marching conservative formulation. For the two-dimensional case, conservation laws are used with body-fitted co-ordinates ξ and η:

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{e}}{\partial \xi} + \frac{\partial \mathbf{f}}{\partial \eta} = 0. \tag{1}$$

The steady state solution is found when the time derivative of the solution vector vanishes. The solution vector and the fluxes normal to the $\xi = \text{const.}$ and $\eta = \text{const.}$ faces are given respectively by

$$\mathbf{u} = J\bar{\mathbf{u}}, \qquad \mathbf{e} = J(\bar{\mathbf{e}}\xi_x + \bar{\mathbf{f}}\xi_y), \qquad \mathbf{f} = J(\bar{\mathbf{e}}\eta_x + \bar{\mathbf{f}}\eta_y) \tag{2}$$

where J is the Jacobian of the inverse mapping. In the Cartesian co-ordinate system the corresponding solution vector and inviscid fluxes are

$$\bar{\mathbf{u}} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \varepsilon \end{bmatrix}, \qquad \bar{\mathbf{e}} = \begin{bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ (\varepsilon + p)u_1 \end{bmatrix}, \qquad \bar{\mathbf{f}} = \begin{bmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ (\varepsilon + p)u_2 \end{bmatrix}, \tag{3}$$

where ε is the total energy ($\varepsilon = p/(\gamma - 1) + 0{\cdot}5\rho(u_1^2 + u_2^2)$) and p and ρ are the pressure and density respectively.

In order to solve equation (1), a cell-centred finite volume scheme with an implicit backward Euler solver[19] for the evolution in time has been used. Since only the steady state solution is required, a scheme with first-order accuracy in time is used and the discretized implicit form of equation (1) is given as

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \mathbf{e}_\xi^{n+1} + \mathbf{f}_\eta^{n+1} = 0. \tag{4}$$

By linearizing the fluxes around the time level n,

$$\mathbf{e}^{n+1} = \mathbf{e}^n + \left(\frac{\partial \mathbf{e}}{\partial \xi}\right)^n \Delta \mathbf{u}^{n+1}, \qquad \mathbf{f}^{n+1} = \mathbf{f}^n + \left(\frac{\partial \mathbf{f}}{\partial \eta}\right)^n \Delta \mathbf{u}^{n+1}. \tag{5}$$

Denoting the correction of the solution vector by $\Delta\mathbf{u}$ ($\Delta\mathbf{u} = \mathbf{u}^{n+1} - \mathbf{u}^n$), a delta formulation of equation (4) can be easily found as

$$\frac{\Delta \mathbf{u}}{\Delta t} + (\mathbf{A}^n \Delta \mathbf{u}_\xi) + (\mathbf{B}^n \Delta \mathbf{u})_\eta = -(\mathbf{e}_\xi^n + \mathbf{f}_\eta^n) = -\mathbf{Res}^n(\mathbf{u}^n), \tag{6}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the Jacobians of the fluxes $\mathbf{e}$ and $\mathbf{f}$ respectively:

$$\mathbf{A}^n = \left(\frac{\partial \mathbf{e}}{\partial \mathbf{u}}\right)^n, \qquad \mathbf{B}^n = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)^n,$$

$$\mathbf{L}^n \Delta \mathbf{u} = (\Delta t^{-1} + \mathbf{A}_\xi^n + \mathbf{B}_\eta^n)\Delta \mathbf{u} = -\mathbf{Res}^n(\mathbf{u}^n). \tag{7}$$

Upwind differencing of the flux vectors is the natural way to reach a diagonally dominant system as well as to introduce numerical dissipation, both crucial for the efficiency and robustness of the solver. For the flux calculations at the faces of the volumes a linear one-dimensional Riemann solver (Godunov approach) has been employed which guarantees the homogeneous property of the Euler fluxes.[19] The mean values of the conservative variables at both sides of the face are used as flow variables at the volume face for the Riemann solver. Depending on the sign of the eigenvalues λ of the local Jacobians $\mathbf{A}$ and $\mathbf{B}$, the conservative variables are extrapolated up to third-order accuracy in the computational space to each face of the volumes (MUSCL-type interpolation). Sensing functions are used to guarantee the monotonic behaviour of the solution, i.e. in shock regions the accuracy of the solution decreases to avoid oscillations. The flow quantities at the faces are defined as the mean value of the left (l) and right (r) states:

$$\mathbf{u}_{i+1/2} = 0{\cdot}5[(1 + \operatorname{sign}\lambda)\mathbf{u}_l + (1 - \operatorname{sign}\lambda)\mathbf{u}_r]. \tag{8}$$

The order of accuracy of the flux difference operator is controlled using an interpolation procedure involving several volumes from both sides of the corresponding volume in each direction, e.g. for the ξ-direction we have

$$\mathbf{u}_l = \frac{1}{e}(a\mathbf{u}_i + b\mathbf{u}_i + b\mathbf{u}_{i-1} + c\mathbf{u}_{i+1} + d\mathbf{u}_{i+2}), \tag{9a}$$

$$\mathbf{u}_r = \frac{1}{e}(a\mathbf{u}_{i+1} + b\mathbf{u}_{i+2} + c\mathbf{u}_i + d\mathbf{u}_{i+1}), \tag{9b}$$

where a, b, c, d and e are properly defined scalar quantities. Thus for first-order accuracy it holds that $e = a = 1$ and $b = c = d = 0$, for second-order accuracy it holds that $e = 2$, $a = 3$, $b = -1$ and

$c = d = 0$, etc. A linear superposition of the state vector computed from equations (9) for different accuracies finally defines the left and right states at the corresponding face as

$$\mathbf{u}_{l,r} = \{[\mathbf{u}^4_{l,r}(1 - c_1) + c_1\mathbf{u}^3_{l,r}](1 - c_2) + c_2\mathbf{u}^2_{l,r}\}(1 - c_3) + c_3\mathbf{u}_{l,r}. \tag{10}$$

The superscripts in equation (10) denote the order of accuracy of the state vector computed from equations (9), c_1 is a user-specified scalar quantity and c_2 and c_3 are sensing functions detecting shocks and spikes respectively, defined as

$$c_2 = \min\left[1, a_1\left(M^2_{\xi\xi}|_i + M^2_{\xi\xi}|_{i+1}\right)\right], \tag{11a}$$

$$c_3 = \min\left|1, a_2\left(M^2_{\xi\xi}|_i + M^2_{\xi\xi}|_{i+1}\right)\right|, \tag{11b}$$

where M is the local second derivative of the Mach number in the ξ-direction and a_1 and a_2 are user-defined scalar quantities. Since the Euler equations are a coupled system, the average state at the cell face, from which inviscid fluxes are obtained, is calculated from the solution of a Riemann problem. With the state vector calculated from equations (8)–(10), the inviscid fluxes are computed directly (flux difference splitting) as

$$\mathbf{e}_{i+1/2} = \mathbf{e}(\mathbf{u}_{i+1/2}). \tag{12}$$

With the divergence of the characteristically extrapolated fluxes on the right-hand side (RHS), equation (7) has to be solved approximately at each time step. In volume (i, j) the variation in $\Delta\mathbf{u}$ with time is calculated as

$$\mathbf{A}\Delta\mathbf{u}_{i-1,j} + \mathbf{B}\Delta\mathbf{u}_{i,j-1} + \mathbf{C}\Delta\mathbf{u}_{i,j} + \mathbf{D}\Delta\mathbf{u}_{i,j+1} + \mathbf{E}\Delta\mathbf{u}_{i+1,j} = \omega\mathbf{Res}^n, \tag{13}$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{E}$ are 4×4 submatrices emerging from a Godunov-like first-order flux-splitting scheme:

$$\mathbf{A} = [0{\cdot}5(c_2 - 1)]\mathbf{A}^+_{i+1/2,j} - \mathbf{A}^+_{i-1/2,j}, \qquad \mathbf{B} = [0{\cdot}5(c_2 - 1)]\mathbf{B}^+_{i,j+1/2} - \mathbf{B}^+_{i,j-1/2},$$

$$\mathbf{C} = \Delta t^{-1} + (1{\cdot}5 - 0{\cdot}5c_2)(\mathbf{B}^+_{i,j+1/2} - \mathbf{B}^+_{i,j-1/2} + \mathbf{A}^+_{i+1/2,j} - \mathbf{A}^+_{i-1/2,j}),$$

$$\mathbf{D} = \mathbf{B}^-_{i,j+1/2} - [0{\cdot}5(c_2 - 1)]\mathbf{B}^-_{i,j-1/2}, \qquad \mathbf{E} = \mathbf{A}^-_{i+1/2,j} - [0{\cdot}5(c_2 - 1)]\mathbf{A}^-_{i-1/2,j},$$

$$\mathbf{Res} = -[\mathbf{e}(\mathbf{u}^n_{i+1/2,j}) - \mathbf{e}(\mathbf{u}^n_{i-1/2,j})] - [\mathbf{f}(\mathbf{u}^n_{i,j+1/2}) - \mathbf{f}(\mathbf{u}^n_{i,j-1/2})];$$

c_2 is used to reduce the accuracy order of the flux-splitting scheme due to discontinuities and is found from equation (11a).

Very large CFL numbers (150–200) can be used, since the true representation of the fluxes of the left-hand side (LHS) Jacobians has been employed. When equation (13) is applied to the entire computational domain, a large block pentadiagonal system emerges which can be solved iteratively. The term ω is an underrelaxation factor which compensates the different spatial order of accuracy between the RHS and LHS of equation (7) and for the present implementation varies from 0·45 to 0·60. Because of the time-marching approach, the solution procedure can be further accelerated by advancing in time with the local optimal time step Δt, keeping the CFL number constant (local time stepping).

Boundary conditions are applied on both sides of equation (7). For the inviscid fluxes, characteristic boundary conditions found from the one-dimensional Riemann solution at the wall and at the freestream boundaries are stored in phantom cell rows in the boundary volumes. Thus the solution method extracts automatically the required information. For the $\Delta\mathbf{u}$ variables, simple boundary conditions are also prescribed at the phantom cells, thus avoiding complex manipulations on the LHS of equation (7).

3. MULTIGRID METHOD

For time-dependent (and time-marching) discretized equations the allowable time step increases with the mesh size according to the CFL condition. Hence, employing fine grids, more computational work per time step and more time steps are required to reach the steady state solution, yielding a quadratic increase in the overall computing cost with the mesh size. For the present multigrid scheme it is necessary to define a sequence of grids in such a way that a coarse volume is constructed by four volumes of the next finer grid level, deleting every other grid line. This cellwise coarsening is essential so that the fine grid fluxes are conserved at the coarser grid levels. Owing to the implicit nature of the solution procedure and the existence of two differential operators in equation (7), the RHS operator on the coarser grids should be transferred from the finest grid level, whereas the LHS operator of equation (13) can be calculated at each grid level. For composite grids the full approximation scheme (FAS) is preferable since it operates at the coarser grid levels with the finest grid variables. Formulating the FAS with the 'alternative point of view',[20] the finest grid is considered as a device to improve the spatial accuracy of the solution, whereas most of the relaxation work is spent at the coarser grid levels. Hence, instead of interpreting the coarser grids as subsidiary grids to smooth the high-frequency error components of the finest grid level, the coarser grid levels are considered as basic grids for the relaxation process, while the finer grids are considered as devices to compute accurate fine-to-coarse multigrid defect corrections (τ). In the present multigrid scheme the solution formulation is made independent of the grid level (coarse or fine) and the type of grid (local or global) by simply adding to the RHS of equation (7) the appropriate fine-to-coarse defect correction. Thus, with the current grid level denoted by n, its next finer level by $n+1$ and its local finest level by N ($N \geqslant n \geqslant 1$; 1 is the coarsest grid level of the domain), the multigrid solution formulation is given as

$$\mathbf{L}_n \Delta \mathbf{u}_n = -\mathbf{Res}_n(\mathbf{u}_n) + \boldsymbol{\tau}_{n+1}^n, \tag{14}$$

where the fine-to-coarse defect correction is

$$\boldsymbol{\tau}_{n+1}^n = \mathbf{L}_n(I_{n+1}^n \Delta \mathbf{u}_{n+1}) - \Sigma_{n+1}^n(\mathbf{L}_{n+1} \Delta \mathbf{u}_{n+1}), \quad \text{with } \boldsymbol{\tau}_{N+1}^N = 0. \tag{15}$$

Because the multigrid cycle coincides with the time step, the time scale will not be considered.

For the coarser grid generation the cellwise coarsening technique has been adopted, meaning that four fine volumes (2×2) are joined together to construct a coarse volume. Maintaining the outer faces of the volumes, flux conservation is achieved easily not only at the coarser grid levels but also at the local subgrid boundaries. However, the coarse grid cell centres are not a subset of the fine ones, so two different restriction operators are required. The restriction operator (I) for the physical variables is the simple average of the four finer volumes:

$$\Delta \mathbf{u}_n = I_{n+1}^n \Delta \mathbf{u}_{n+1} = \frac{\sum \Delta \mathbf{u}_{n+1}}{4}. \tag{16}$$

The restriction operator (Σ) for the generalized residuals **Res** and $\boldsymbol{\tau}$ is the algebraic summation of the residuals of the corresponding finer volumes. The fluxes of the inner common fine grid faces are cancelled, so the finest grid fluxes are restricted to the coarser grid levels without loss:

$$\mathbf{Res}_n = \Sigma_{n+1}^n \mathbf{Res}_{n+1}. \tag{17}$$

For the coarse-to-fine direction of the multigrid cycle, neither Euler equation solutions nor relaxation sweeps are required. Therefore only the $\Delta \mathbf{u}$ variables are stored for all grid levels and prolongated from the coarse to the fine grid levels using the standard FAS prolongation form

$$\Delta \mathbf{u}_{n+1} = \Delta \mathbf{u}_{n+1} + \mathrm{II}_n^{n+1}(\Delta \mathbf{u}_n - I_{n+1}^n \Delta \mathbf{u}_{n+1}). \tag{18}$$

For the prolongation operator (II) simple injection instead of bilinear interpolation has been used to facilitate the prolongation procedure in composite grids.

In terms of the multigrid strategy, two important issues should be highlighted. The first concerns the number of relaxation sweeps at each grid level. Owing to the adopted implicit solution scheme, the convergence of the multigrid scheme improves when fewer relaxation sweeps are performed on the finer grids and more relaxation work is spent at the coarser grid levels. Thus the number of sweeps at each level increases proportionally to the depth of the grid level.[20] Secondly, for the coarse-to-fine direction, no relaxation sweeps are performed except for the prolongation steps of the correction vector $\Delta\mathbf{u}$ (equation (18)). Using this kind of V-cycle, insignificant extra storage ($\leqslant 5\%$) is demanded by the entire multigrid implementation. Except for the correction vector $\Delta\mathbf{u}$, no other variables and fluxes are required in the coarse-to-fine direction, so for the coarse grid variable storage the same positions as for the finest grid variable storage can be used. For the relaxation scheme, although collective Gauss–Seidel relaxation in lexicographic order behaves satisfactorily for the single-grid code, its symmetric variation has been employed. The symmetric collective Gauss–Seidel relaxation scheme improves the smoothing properties, avoiding possible grid alignment. Thus the robustness and efficiency are increased without implementing a computationally expensive and complex scheme such as line relaxation.

4. SOLUTION ERROR ESTIMATION

To identify the regions of the computational domain with excessive solution errors, a reliable error sensor is required. To evaluate the solution error, two approaches essentially exist: the first involves the physically based information on the problem, i.e. solution gradients, while the second, numerically motivated, is the evaluation of the discretization error. The former approach may implicate the refinement process to reduce errors that have no influence on the global solution. In contrast, the evaluation of the discretization error indicates errors that can be confronted by the refinement procedure. An estimation of the discretization error is given by the truncation error[12,13] $(\mathbf{t})$, which on the basis of a Taylor expansion is defined by

$$\mathbf{t}_n = \mathbf{Q}_n\mathbf{u}, \tag{19}$$

where $\mathbf{Q}$ is the differential operator, $\mathbf{u}$ is the physically correct solution and n is the typical mesh size of the finest grid level N. Guided by the physical interpretation of the truncation error concept and provided that the solution is smooth at the domain, Richardson extrapolation can extend the validity of equation (19) to practical problems. Thus the difference in truncation error between two consecutively fine grid levels N and $N-1$ can be used to approximate the solution error:

$$\mathbf{t}_N^{N-1} = \mathbf{Q}_N\mathbf{u}_N - \mathbf{Q}_{N-1}\mathbf{u}_{N-1}. \tag{20}$$

Because of the implicit solution procedure, for the choice of the differential operator $\mathbf{Q}$ two possibilities essentially exist, namely the operators $\mathbf{L\Delta u}$ and $\mathbf{Res(u)}$ of equation (7). Adopting the operator $\mathbf{L\Delta u}$ equation (20) practically coincides with equation (15) of the multigrid defect correction τ which is directly provided by the multigrid solution. However, owing to the implicit solution procedure, the operator $\mathbf{L\Delta u}$ includes more relaxation errors and is calculated with worse accuracy than the operator $\mathbf{Res(u)}$.[21] Therefore the solution error evaluation for the grid level $N-1$ is given by

$$\mathbf{t}_N^{N-1} = \Sigma_N^{N-1}\mathbf{t}_N - \mathbf{t}_{N-1} = \Sigma_N^{N-1}\mathbf{Res}_N(\mathbf{u}_N) - \mathbf{Res}_{N-1}(\mathbf{I}_N^{N-1}\mathbf{u}_N), \tag{21}$$

whereas for a fully converged solution equation (21) reduces to

$$\mathbf{t}_N^{N-1} = -\mathbf{Res}_{N-1}(\mathbf{I}_N^{N-1}\mathbf{u}_N). \tag{22}$$

The proposed error sensor requires additional work of only one-fourth of a simple flux calculation and does not demand a totally converged solution ($\mathbf{Res}_M(\mathbf{u}_N) \neq 0$) as it converges from the initial time steps to its steady state value. Since the truncation error sensor is a vector, to convert it to a scalar variable the Euclidean norm has been adopted, because it behaves similarly to the pressure error of the solution.[21]

In shock regions where the solution is discontinuous, the Richardson extrapolation scheme is not valid any more. As can be shown,[21] in these regions the calculated truncation error is very large and independent of the mesh size. Thus in the automatic adaptive procedure the regions with discontinuous solutions are always marked for refinement.

5. COMPOSITE GRID STRUCTURE AND SOLUTION

For the achievement of the most accurate solution with the minimum amount of work, several grid adaptive techniques and structures have been proposed. In general the composite grid structure has many advantages by enabling the decomposition of a globally non-uniform grid into a union of locally uniform subgrids.[2] On the other hand, subgrid uniformity is essential to ensure multigrid efficiency using simple integration routines (similar to the single-grid solver). Moreover, in the computational domain, when the subgrids are restricted to rectangles and the grid refinement ratio is to 2 between neighbouring subgrids,[9–11,22] considerable simplifications to the data structure and the interface are attained.

5.1. Multigrid composite structure

For the dynamically adaptive multigrid strategy a modified full multigrid scheme has been developed. Starting the solution procedure with a global coarse grid of acceptable grid resolution after solution convergence (or after a fixed amount of work), the truncation error is calculated and the solution error is predicted. In regions where the prediction of the error is above a given threshold, the corresponding volumes are flagged and grouped into rectangular blocks. Then the domain is decomposed into the appropriate subgrids and only those which contain flagged volumes are advanced to the next grid level, injecting also the coarse grid solution to the just refined subgrids. The refinement procedure continues until the entire computational domain presents local truncation errors below the preset threshold. Clearly this strategy has the benefit of a continuous iterative procedure without wasting computational work on calculations that will not be used in the next mesh refinement step. Taking advantage of the most accurate available solution, the proposed grid adaptive strategy converges fast to the most efficient solution.

5.2. Internal boundary conditions

A considerable advantage of the composite grid consisting of rectangular subgrids is the use of similar integration routines as for global grids. However, for the implicit solution schemes of hyperbolic equations, extra requirements at the artificial boundaries are essential, since the accuracy and convergence rates of the global grid solution should be maintained. To reduce error propagation at the artificial boundaries, specific artificial boundary conditions must hold. Since two differential operators are used (equation (7)), different interface solution techniques for each operator should be considered. The operator $\mathbf{Res}(\mathbf{u})$ is responsible for maintaining the accuracy of the solution and ensuring flux conservation at the interfaces, whereas to maintain the convergence rate of the global solution the flux-splitting operator $\mathbf{L}\Delta\mathbf{u}$ should be modified.

Initially the fluxes on the subgrid of the finest grid level should be calculated. For all the subgrids exactly the same integration routine has been used. To cope with the grid non-uniformity at the interfaces, the finer subgrids are extended into the neighbouring coarser subgrids using fictitious zones

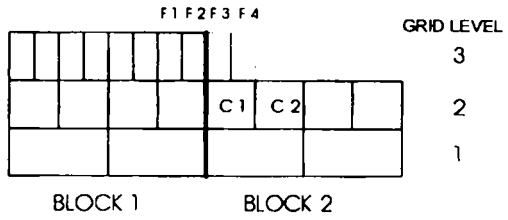


Figure 1. One-dimensional analogue of the construction of the fictitious fine volumes F3 and F4 inside the coarse grid block 2. At the coarser multigrid levels, blocks 1 and 2 form a uniform block (superblock)

with a width of two fine volumes. Thus the flux calculation at the initial interfaces is done as if the fine subgrid was uniform (Figure 1). Obviously the key to maintaining the accuracy of the solution is the correct calculation of the fictitious fine grid volumes. For this task, linear, bilinear or even quadratic interpolation techniques fail, since the wave propagation of the solution is not considered. The most suitable extrapolation procedure for the correct calculation of the fictitious volumes is the same one as used by the global solution scheme (equations (9) and (10)). The proposed extrapolation strategy at the intergrid boundaries is depicted in Figures 1 and 2. To calculate the fine grid fluxes (block 1) at the interface, the fictitious fine volumes F3 and F4 (Figure 1), which correspond to the coarse volume C1, must be calculated first. For this calculation the same MUSCL-type characteristic extrapolation scheme has been employed involving the coarse volumes C1 and C2 and the fine ones F1 and F2. Using equations (9)–(11) the left and right states of volume C1 are considered as the new states F3 and F4; hence the fine grid fluxes at the interface are calculated straightforwardly. It must be highlighted that no interaction of the corresponding row with other rows is allowed, e.g. for the flux calculation at boundary face 1 (Figure 2) the same pair of coarse volumes (C1 and C2) is used in conjunction with the fine volumes F1 and F2 to define states F3 and F4.

After calculating the fluxes of the finest subgrid (block 1 in Figure 1), the interface fluxes of the neighbouring coarser subgrids (block 2) can be calculated explicitly using conservation of fluxes. According to the multigrid restriction operator for the residuals, flux conservation across the interfaces is achieved by addressing the summation of the fluxes of a pair of fine volume faces to their adjacent coarse volume face (Figure 2).

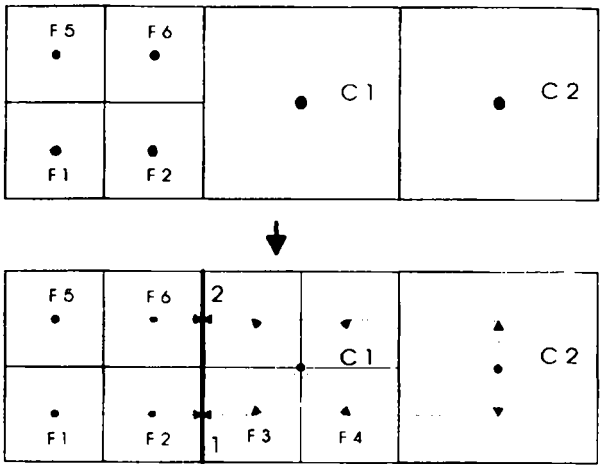


Figure 2. Transformation of the non-uniform grid to a fictitious uniform one. For the calculation of the fluxes at face 1, MUSCL interpolation of the coarse volumes C1 and C2 is used to produce the fictitious fine volumes F3 and F4

Concerning the relaxation procedure, no modifications to the flux-vector-splitting scheme and relaxation solution scheme at the subgrid interfaces are required,[14,22] since owing to the multigrid algorithm the relaxation solution procedure refers only to a uniform grid. The relaxation scheme sweeps only subgrids that are at the same grid level either originally or as a result of restriction by the multigrid process. For example, block 1 in Figure 1 is relaxed first when the control of the multigrid cycle is at the third grid level, whereas both blocks 1 and 2 are relaxed when the multigrid control is at the second grid level (Figure 1).

6. THE PARALLEL BAM METHOD

In general the parallelization of dynamically adaptive multigrid schemes has faced several problems.[15,16,17] However, for the present method, owing to the rectangular shape of the subgrids of the composite grid, parallelization can be straightforward when ideas from domain decomposition theory are introduced. For example, one possible technique could be to consider one or more subgrids of the composite grid continuously attached to a single processor for the entire multigrid cycle (from the finest to the coarsest level), while communication between subgrids of different processors is carried out only once per multigrid cycle (vertical communication mode[18]). Using this method, load balancing is very difficult to achieve, so reduced performance would be expected.

A different approach, which has been adopted in the present study, is to modify the block structure of the composite grid only at the relaxation step, employing domain decomposition techniques with load-balancing criteria. Taking advantage of the rectangular shape of the subgrids, it is possible to split the relaxation work (most of the calculations) equally among the available processors. Following this approach, which is based on the FAC method,[16] at each multigrid level of the relaxation step the subgrids of mesh density equal to or higher than the current multigrid level are recomposed to form larger subgrids (superblocks) of rectangular shape where possible. The superblocks are redecomposed to blocks, with their size depending on the number of processors, and finally each block is attached to each processor for the relaxation step. Referring only to computing machines with a small or moderate number of processors, the proposed partitioning strategy has the advantage of perfect load balancing independently of the dynamically adaptive structure of the composite grid. Furthermore, to avoid any data interdependences, the same Gauss–Seidel point relaxation scheme has been used everywhere except at the new block interfaces, where the relaxation scheme changes to Jacobi type.

7. RESULTS

In order to verify the accuracy and validate the efficiency of the proposed method, two popular transonic inviscid test cases have been investigated. The first case is an NACA-0012 aerofoil at Mach 0·80 and an angle of attack of 1·25°, while the second case is an RAE-2822 aerofoil at Mach 0·73 and 2·79°. A work unit is defined as the CPU time required for a global finest grid relaxation sweep in lexicographic order, while for a single-grid run one time step costs four work units (four relaxation sweeps per time step).

Starting from a two-grid global multigrid scheme with 64 × 14 volumes at the finest grid level, two additional grid refinement levels are allowed for both test cases. For the convergence criterion the Euclidean norm of the correction vector is employed. For the first test case (NACA-0012) the computed truncation error contours (Figure 3) and the pressure error contours (Figure 4) are depicted. The comparison between the computed truncation and pressure errors shows very similar results. Taking into consideration the truncation error prediction, the adaptive mesh generation procedure is shown in Figure 5. Starting from a global coarse grid (64 × 14; Figure 5(a)), a new composite grid of four subgrids is generated (Figure 5(b)) and after the last truncation error prediction a composite grid

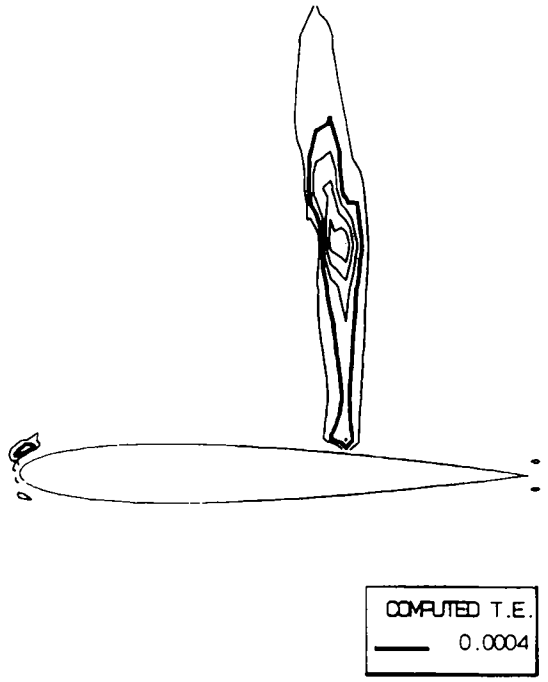


Figure 3. Computed truncation error contours for the finer grid level (case 1)

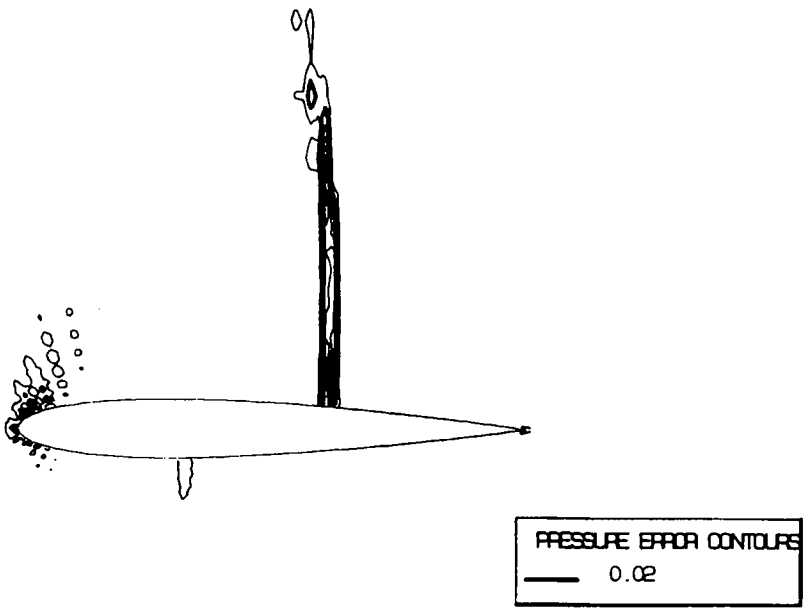
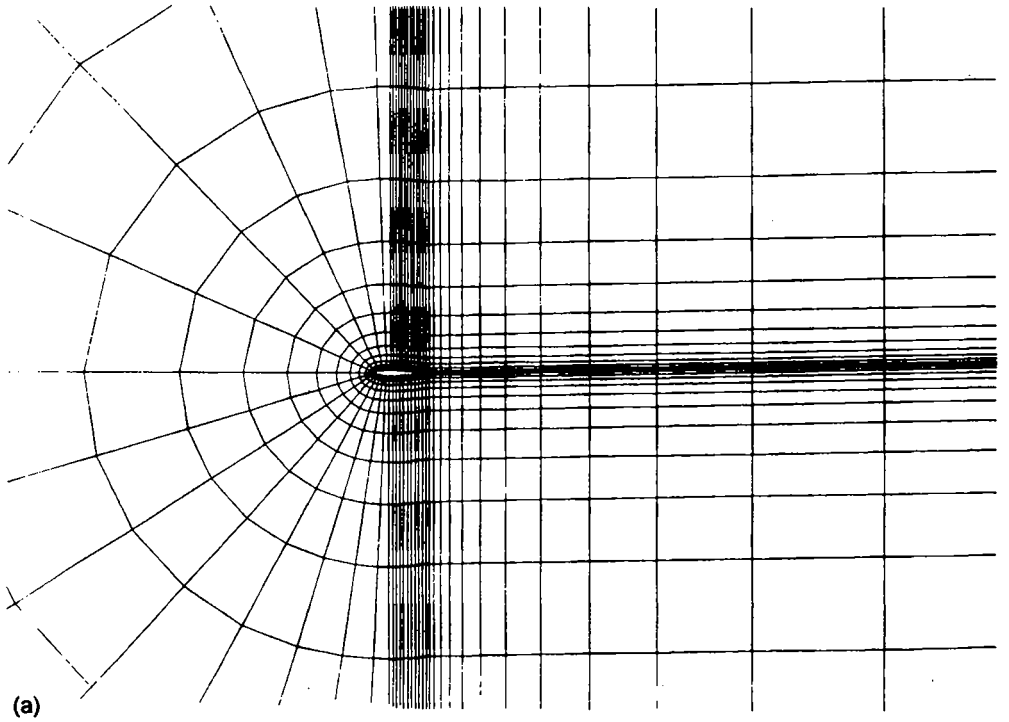
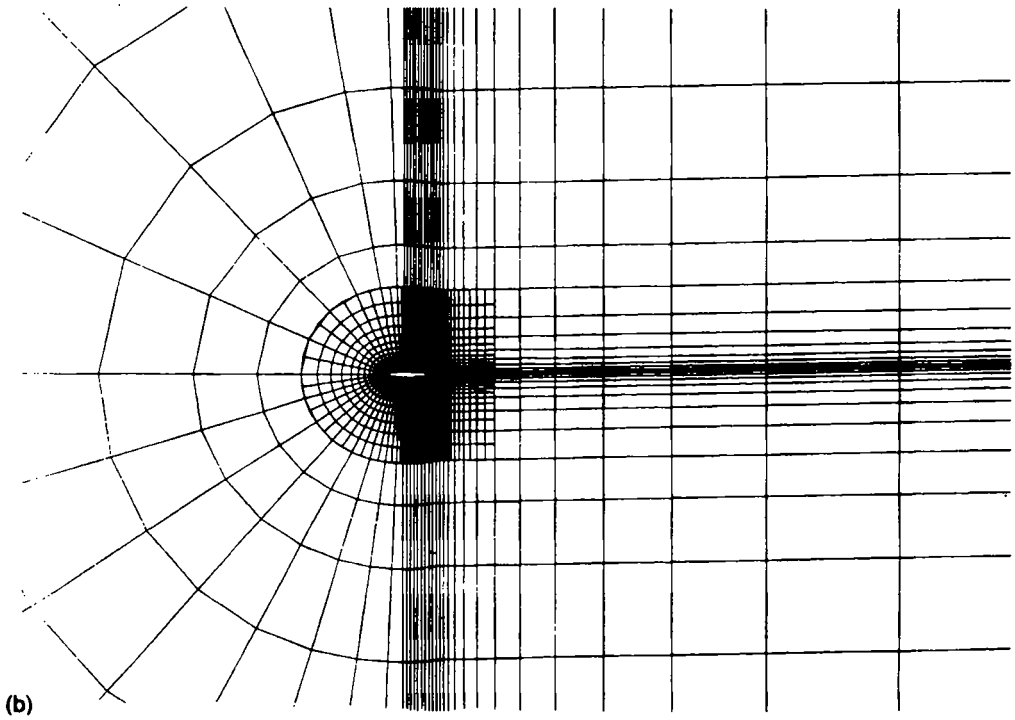


Figure 4. Total pressure error contours for the finer grid level (case 1)



(a)



(b)

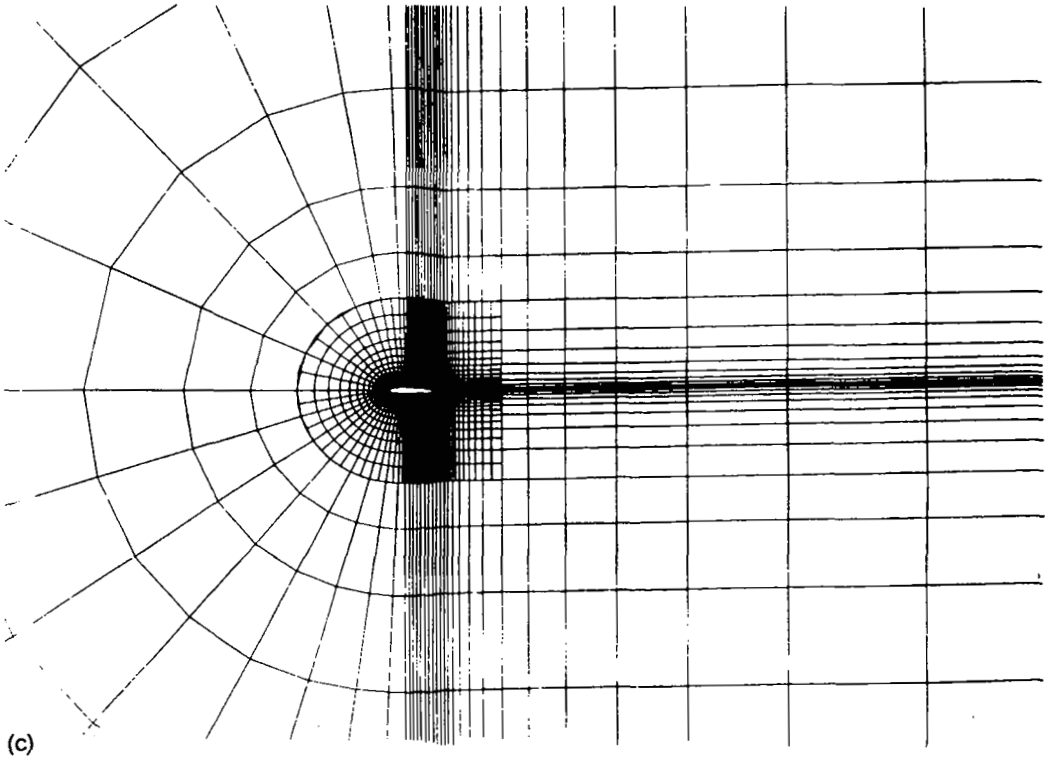


Figure 5. Adaptive mesh generation steps of the BAM method (case 1): (a) the initial (64 × 14) mesh; (b) one local refinement creates four subgrids (one subgrid at the finer grid level); (c) the final composite mesh with nine subgrids (two subgrids at the finest and four at the finer levels)

consisting of nine subgrids at three different grid levels is finally generated as can be seen in Figure 5(c). In a closer view of the aerofoil the Mach contours together with the composite grid are depicted in Figure 6. By adopting the proposed method, very accurate results can be obtained in comparison with the global grid solutions as depicted in Figure 7, where the Mach distributions along the aerofoil are shown for two global grids (256 × 56 and 128 × 28) and one adaptive composite grid. The comparison of Mach distributions along the aerofoil shows that in regions of similar mesh size the solutions from the global grid and the local refinement practically coincide. In Figure 8 the great efficiency of the BAM method with respect to the single-grid and global multigrid schemes is clearly demonstrated, where 19-fold and fourfold accelerations are achieved respectively. The final adaptive grid solution requires 4·5 times fewer volumes (from 14,336 to 3200) for practically the same accuracy as with the globally refined grid (0·35% discrepancy in the computed lift coefficient C_L).

Similar efficiency has been achieved for the second test case. The final domain decomposition into nine subgrids takes place after 50 times steps on coarser grid levels. Using the same truncation error threshold as in the previous test case, a 17-fold acceleration has been achieved with respect to the single grid and a 4·7-fold reduction in the number of volumes (from 14,336 to 3056) for practically the same accuracy (C_L discrepancy 0·2%). The convergence histories of the error reduction and the lift coefficient are shown in Figure 9, while in Figure 10 the final composite grid together with the isomach contours is depicted. It is important that throughout the solution process the multigrid convergence rates are maintained while the overhead for the interface computations is negligible, e.g. the overhead is only 2% for a nine-block structure with respect to an equivalent global grid.

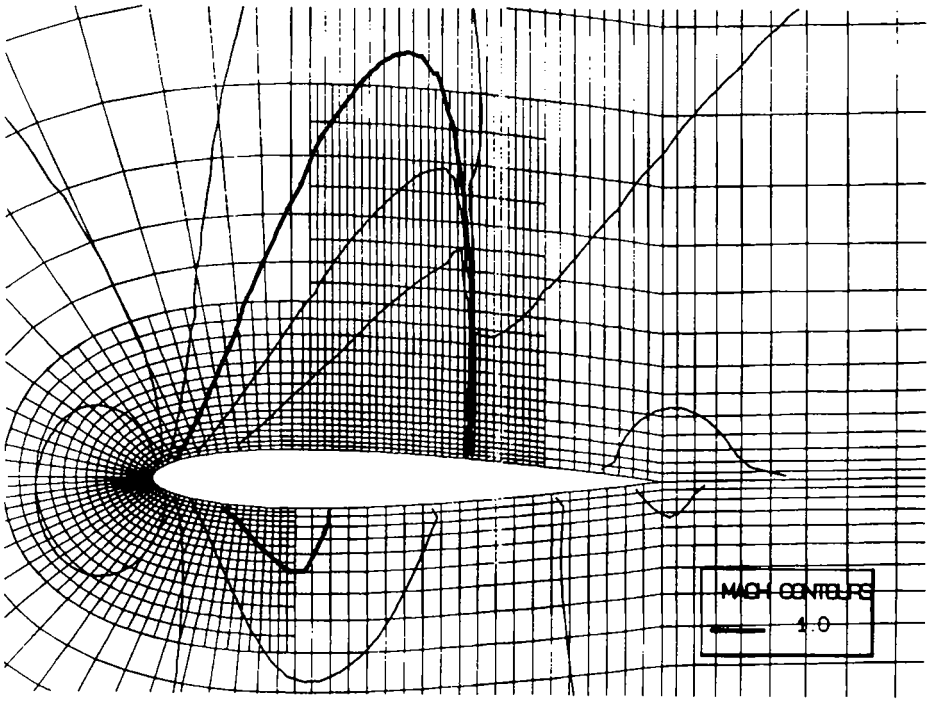


Figure 6. Closer view of the composite mesh together with the Mach contours (case 1)

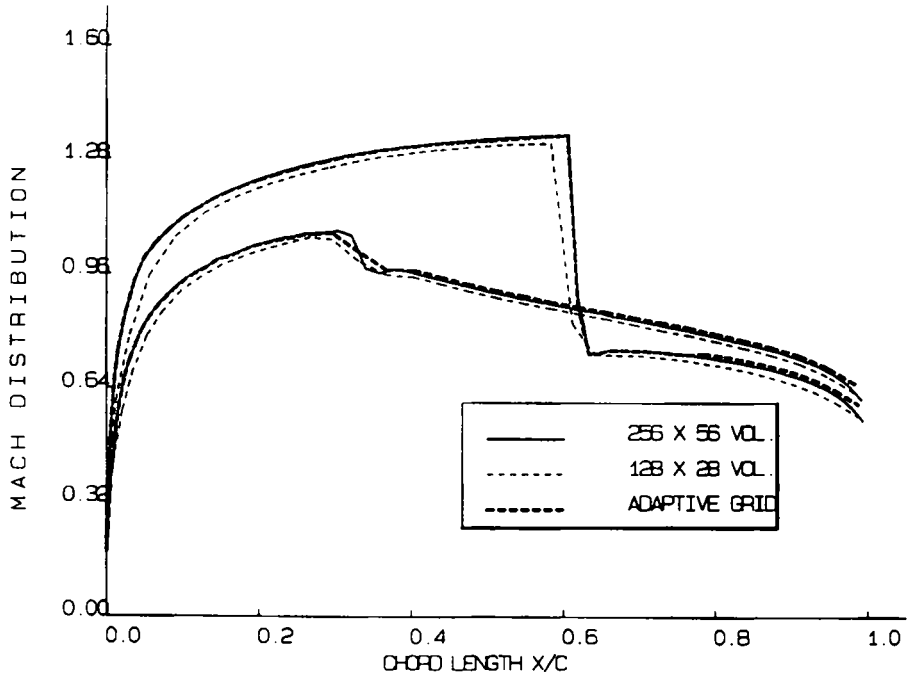


Figure 7. Mach distribution along the aerofoil for two global grids (256 × 56 and 128 × 28) and an adaptive grid (case 1)

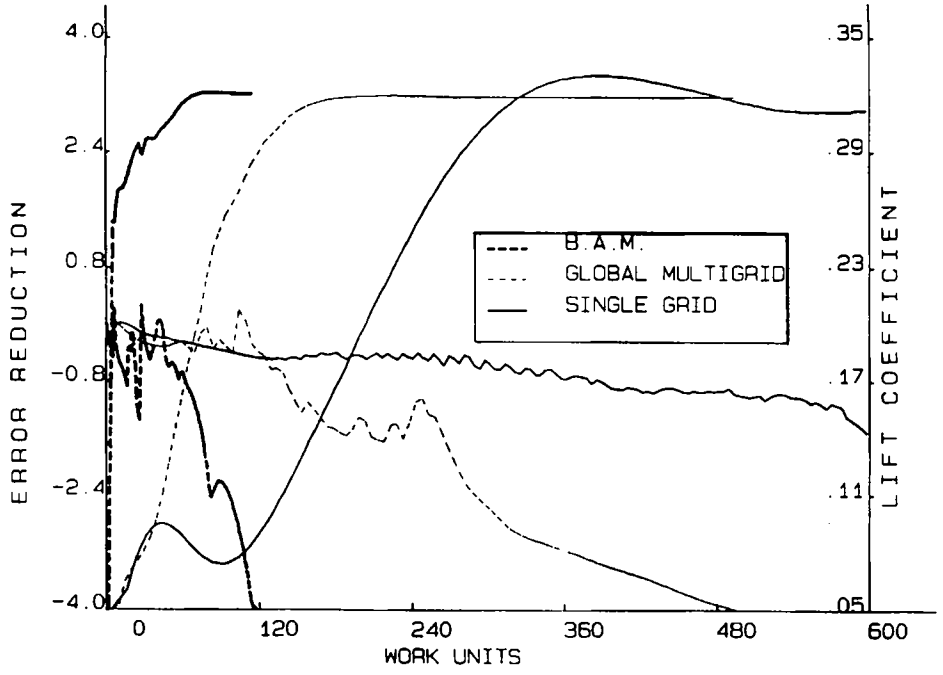


Figure 8. Convergence histories of the lift coefficient and the logarithmic Euclidean error for the single grid, the global multigrid and the BAM method (case 1)

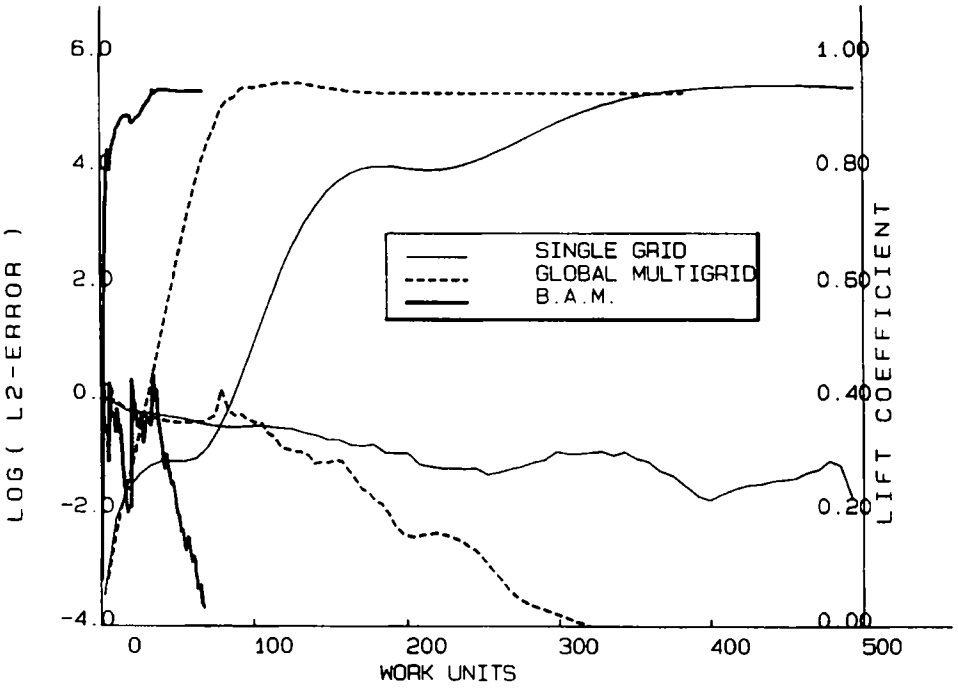


Figure 9. Convergence histories of the lift coefficient and the logarithmic Euclidean error for the single grid, the global multigrid and the BAM method (case 2)

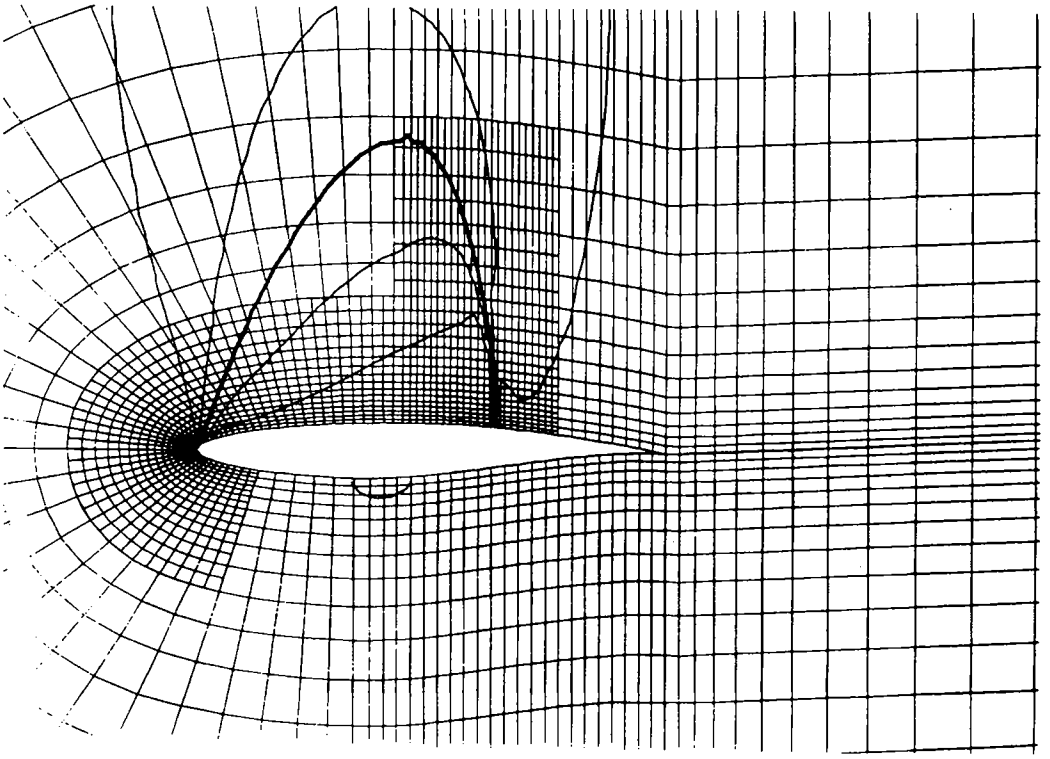


Figure 10. Closer view of the composite mesh together with the Mach contours (case 2): ————, Mach 1

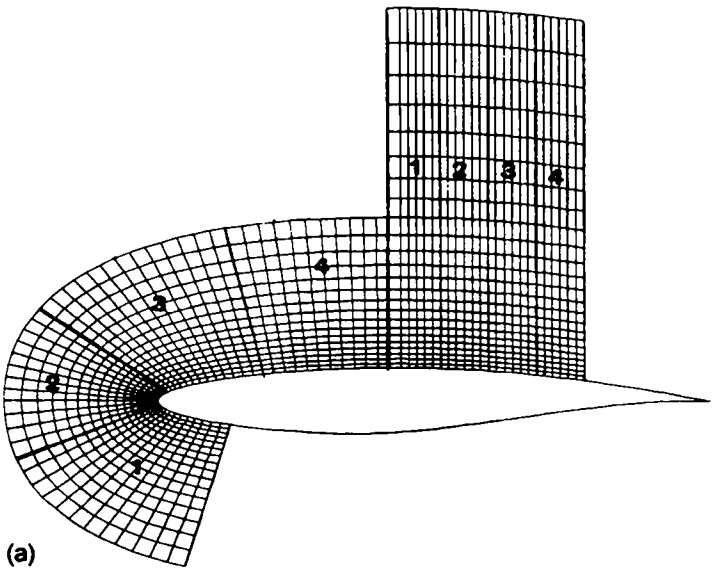
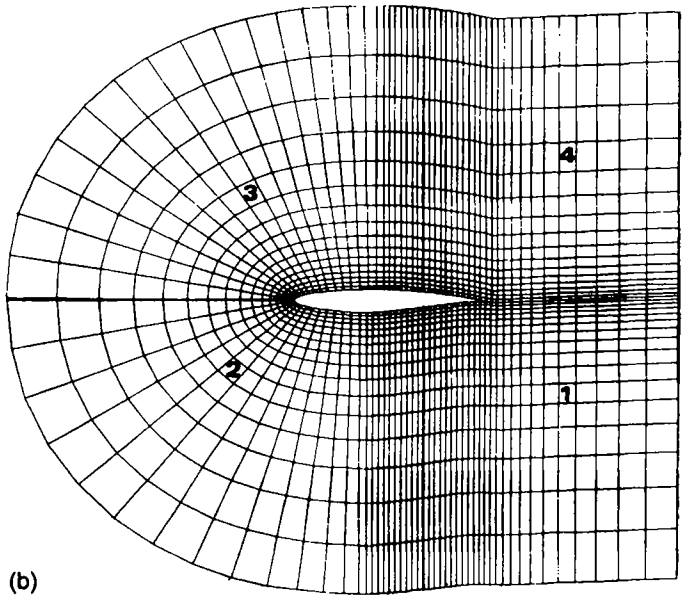


Figure 11(a). For description see over.



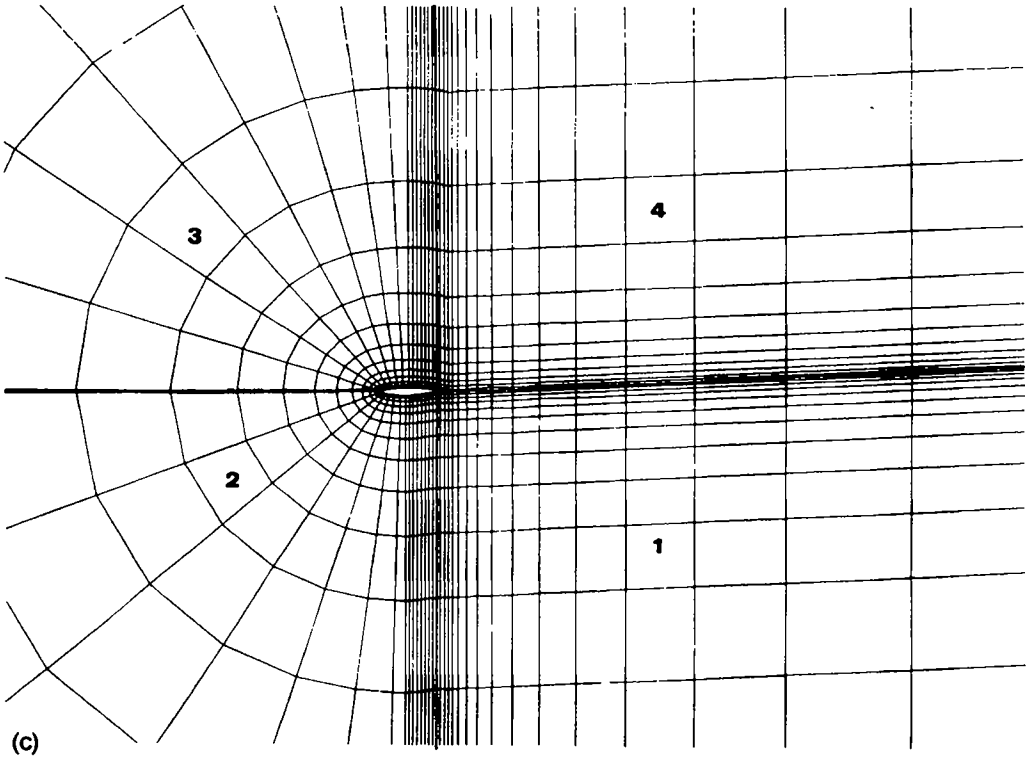


Figure 11. Parallelization—the decomposition of the superblocks depending on the multigrid level (case 2): (a) the two finest subgrids are decomposed into four blocks each (the subgrids cannot form one rectangular superblock); (b) at the coarser multigrid level, six subgrids form a single superblock; (c) at the next coarser level, nine subgrids form a superblock which is a global grid (the numbers denote the blocks and the corresponding processor)

Finally, to test the performance of the proposed BAM method on parallel computers, a four-processor machine has been used. The domain decomposition procedure proposed is shown in Figure 11 for the second test case. Solving the final composite grid (nine subgrids at three different grid levels) at the finest multigrid level, each finest subgrid is divided into four blocks which are assigned to each of the processors (Figure 11(a)). When the multigrid cycle moves to the next coarser level, at the relaxation step the four subgrids of the current grid level together with two subgrids of the finest level are joined together to form the superblock depicted in Figure 11(b). Then this superblock is decomposed into four blocks which are relaxed in parallel. Finally, at the other grid levels of the multigrid cycle the superblocks formed are global grids (they cover the entire domain) and again they are divided into four equal sublocks (Figure 11(c)). The convergence histories are depicted in Figure 12, where the comparison between the serial code and the parallel code for a single processor (where the superblocks are still divided into four blocks) shows that although communication among blocks is reduced, the convergence rate is presented. Clearly, when four processors are used, the computer time is halved, though for finer grids the performance is expected to increase more. In Figure 12 it can be seen that with the cost of only 20 work units and one-quarter of the number of volumes of a global grid an accurate solution of the flow can be provided.

8. CONCLUSIONS

The great advantages of the block adaptive multigrid (BAM) method are exhibited. The incorporation of numerous efficient schemes into the BAM method makes the ultimate target of solving complex problems in just a few work units feasible. At the same time the robustness, simplicity and accuracy of the single-grid code are maintained with the new method. Although the basic features of the BAM

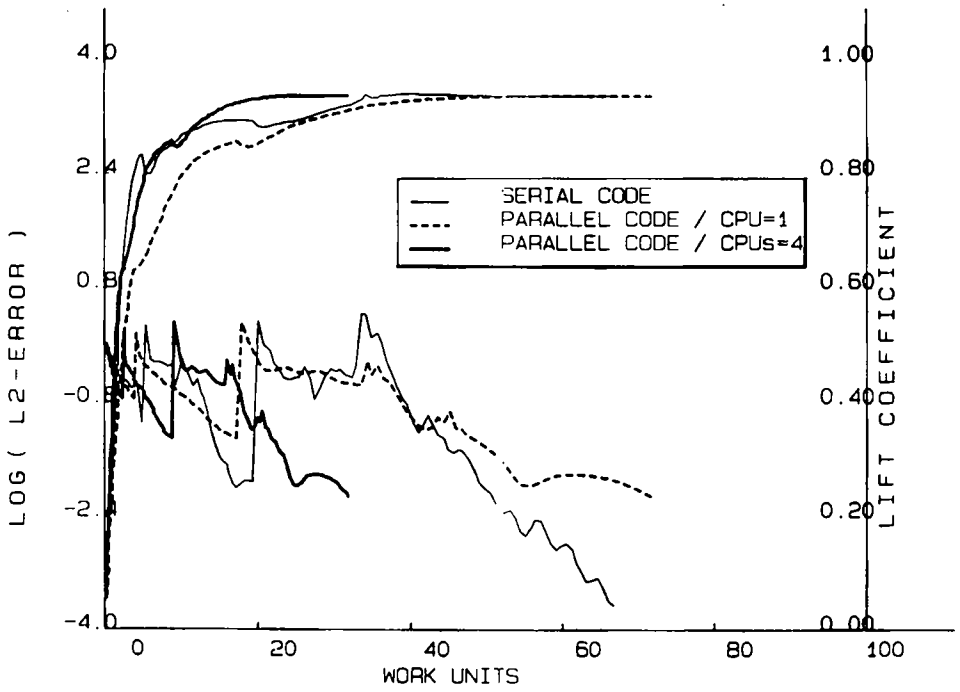


Figure 12. Convergence histories of the lift coefficient and the logarithmic Euclidean error for the serial code, the parallel BAM method on one processor (but with simulation for four processors) and the parallel BAM method on four processors (case 2)

method have been determined and verified, some issues remain to be settled. The first is the development of a data structure that will handle more efficiently the block structure of the composite grid. The second issue is to combine the truncation error prediction with other solution error techniques for more accurate predictions of viscous and hypersonic flows.

The extension to viscous and three-dimensional problems is straightforward, though semi-coarsening multigrid can also be included especially for hypersonic and turbulent flows. On the other hand, to improve the grid adaptation capabilities, a combination of the present method with a moving grid point scheme should also be considered, since grid alignment towards certain flow features is essential in some flow problems. Additionally, the implementation of the BAM method for other solution algorithms and equations is foreseen, since the BAM method has been designed within the general concept of the finite volume method.

REFERENCES

1. A. Brandt, 'Multi-level adaptive solutions to boundary-value problems', *Math. Comput.*, **31**, 333–390 (1977).
2. S. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1989.
3. C. Liu, 'The finite volume element (FVE) and fast adaptive composite grid methods (FAC) for the incompressible Navier–Stokes equations', *Proc. 4th Copper Mountain Conf. on Multigrid Methods*, SIAM, Philadelphia, PA, 1989, pp. 319–337.
4. W. Joppich, 'A multigrid algorithm with time-dependent, locally refined grids for solving the nonlinear diffusion equation on a nonrectangular geometry', *Proc. 3rd Eur. Conf. on Multigrid Methods*, ISNM Vol. 98, Birkhauser, Basle, 1991, pp. 241–252.
5. D. M. Smith and A. D. Gosman, 'An application of multigrid with local grid refinement to fluid flow calculations,' *Prelim. Proc. 5th Copper Mountain Conf. on Multigrid Methods*, 1991.
6. K. Srinivasan and S. G. Rubin, 'Adaptive multigrid domain decomposition solutions of the reduced Navier–Stokes equations', *Proc. Domain Decomposition Conf.*, SIAM, Philadelphia, PA, pp. 586–596, 1992.
7. M. C. Thompson and J. H. Ferziger, 'An adaptive multigrid technique for the incompressible Navier–Stokes equations', *J. Comput. Phys.*, **82**, 94–121 (1989).
8. J. F. Dannenhoffer II, 'A comparison of adaptive-grid redistribution and embedding for steady transonic flows', *Int. j. numer. methods eng.*, **32**, 653–663 (1991).
9. M. J. Berger and A. Jameson, 'Automatic adaptive grid refinement for the Euler equations', *AIAA J.*, **23**, 561–568 (1985).
10. M. J. Berger and P. Colella, 'Local adaptive mesh refinement for shock hydrodynamics;, *J. Comput. Phys.*, **82**, 64–84 (1989).
11. N. Pantelelis 'The block adaptive multigrid method applied to the solution of the Euler equations', *Proc. 6th Copper Mountain Conf. on Multigrid Methods*, NASA CP-3224, 1993, pp. 465–480.
12. K. Y. Fung, J. Tripp and B. Goble, 'Adaptive refinement with truncation error injection', *Comput. Methods Appl. Mech. Eng.*, **66**, 1–16 (1987).
13. Y. N. Jeng and J. L. Chen, 'Truncation error analysis of the finite volume method for a model steady convective equation', *J. Comput. Phys.*, **100**, 64–76 (1992).
14. M. M. Rai, 'Conservative treatment of zonal boundaries for Euler equation calculations', *AIAA Paper 84-0164*, 1984.
15. M. Lemke, K. Witsch and D. Quinlan, 'An object-oriented approach for parallel self adaptive mesh refinement on block structured grids', *Proc. 6th Copper Mountain Conf. on Multigrid Methods*, NASA CP-3224, 1993, pp. 345–360.
16. M. A. Heroux and J. W. Thomas, 'TDFAC: a composite grid method for elliptic equations', *Proc. 4th Copper Mountain Conf. on Multigrid Methods*, SIAM, Philadelphia, PA, 1989, pp. 273–285.
17. H. Sbosny, 'Parallel multigrid methods on composite meshes', *Proc. 5th Int. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM Philadelphia, PA, 1992, pp. 401–408.
18. Y. Yadlin and D. A. Caughey, 'Block multigrid implicit solution of the Euler equations of compressible fluid flow', *AIAA J.*, 712–719 (1991).
19. A. Eberle, 'Characteristic flux averaging approach to the solution of the Euler equations', *VKI Lecture Ser.*, (1987).
20. A. E. Kanarachos and N. G. Pantelelis, 'A multigrid scheme for the implicit solution of the compressible flow equations', *J. Comput. Mech.*, **14**, 235–248 (1994).
21. A. E. Kanarachos, N. G. Pantelelis and I. P. Vournas, 'Multigrid methods for the acceleration and the mesh adaptation of the transonic flow problem', in *Numerical Methods in Fluid Mechanics*, Vol. 44, Vieweg, Braunschweig, 1993, pp. 311–331.
22. A. E. Kanarachos and N. G. Pantelelis, 'Block full multigrid adaptive scheme for the compressible Euler equations', *Proc. 13th Int. Conf. on Numerical Methods in Fluid Dynamics*, LNP Vol. 424, Springer, Berlin, 1993, pp. 265–269.