Journal of Computational Physics 229 (2010) 3523–3542

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Euler calculations with embedded Cartesian grids and small-perturbation boundary conditions

W. Liao [a,b,*], E.P.C. Koh [a], H.M. Tsai [a], F. Liu [c]

[a] Temasek Laboratories, National University of Singapore, Kent Ridge Crescent, Singapore 117411, Singapore
[b] Department of Mathematics and Statistics, Old Dominion University, Norfolk, VA 23529, United States
[c] Department of Mechanical and Aerospace Engineering, University of California, Irvine, CA 92697, United States

ARTICLE INFO

Article history:
Received 17 April 2009
Received in revised form 10 October 2009
Accepted 13 January 2010
Available online 18 January 2010

Keywords:
Cartesian grid
Embedded multigrid
Surface boundary
Least-squares approximation
Small-perturbation method

ABSTRACT

This study examines the use of stationary Cartesian mesh for steady and unsteady flow computations. The surface boundary conditions are imposed by reflected points. A cloud of nodes in the vicinity of the surface is used to get a weighted average of the flow properties via a gridless least-squares technique. If the displacement of the moving surface from the original position is typically small, a small-perturbation boundary condition method can be used. To ensure computational efficiency, multigrid solution is made via a framework of embedded grids for local grid refinement. Computations of airfoil wing and wing-body test cases show the practical usefulness of the embedded Cartesian grids with the small-perturbation boundary conditions approach.

1. Introduction

With complex configurations such as an aircraft with stores dealt with by structured, unstructured or hybrid grid methods, very significant effort is needed to create the initial grid and subsequently ensure the fidelity of the moving grid. The task of grid generation is to create just enough grid points to adequately resolve the significant flow features present. In the context of body-conformal curvilinear grid, generating good-quality grids can be challenging involving an iterative process where substantial human intervention is necessary. Inherently there are conflicting requirements in grid resolution and computational efficiency. Such requirements can give rise to grid skewness that negatively impacts the solvers accuracy and convergence. Though conceived to better handle complex geometries, the unstructured grid approach also needs considerable effort to ensure grid quality and careful prior surface grid preparation.

In contrast, geometric complexities have no particular effects on Cartesian grid quality. Cartesian grids devoid of any inherent skewness can be readily created. Non-body-conformal grids in the region near the surface are created by the solid boundary cutting through the Cartesian volume grids. The challenge is in the implementation of solid wall boundary conditions to achieve the necessary accuracy but the process can be automated using suitable algorithms. Modifying the equations along these cut cells is necessary. In addition, by using Cartesian grids one looses control of the grid resolution in the vicinity of the body. For instance, excessive mesh refinement is needed near curved boundaries such as the leading edges of wings. Nesting or embedment of Cartesian grids used in an overlapping manner for local refinement can alleviate this problem. When this embedded mesh is used in the form of a multigrid computation, rapid convergence rate for the solution can

* Corresponding author. Address: Department of Mathematics and Statistics, Old Dominion University, Norfolk, VA 23529, United States. Tel.: +1 757 6836001x5035; fax: +1 757 6833885.
E-mail addresses: wliao@odu.edu (W. Liao), hm_tsai@science.edu.sg (H.M. Tsai), fliu@uci.edu (F. Liu).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.01.014

be achieved [1]. For these advantages, the CFD community has invested considerable attention to Cartesian methods [2–7]. The method is practical for inviscid flows, but is not really suitable for viscous flows unless it is at low Reynolds numbers.

Moreover, Cartesian grid method has the advantage of using significantly less computational resources. Compared to using body-conforming grids, terms involving grid transformations are not required in the computation. When compared to unstructured grids, grid connectivity information is obvious and thus incurs no additional storage requirements. The relative simplicity of structured Cartesian grid based solvers also means simpler and more efficient iterative schemes that exploit the structured nature of the problem, which can enhance the overall computational efficiencies. The computational efficiencies and the great ease in grid generation makes Cartesian grid a compelling approach.

Current computations related with moving boundaries typically take a direct approach involving deformable meshes. While efficient moving grid methodology is available (see Tsai et al. [8]), they may not be sufficiently robust to avoid degenerate grids when it comes to modeling complex geometries. The use of non-deforming Cartesian grid can by-pass the problems associated with generating a new grid and the need to project the solution onto this new grid at each time step. The direct approach would be to work out the new boundary conditions arising from the moving surface relative to the stationary Cartesian grid. However, the procedures to search and determine new cut cells at each time step make this computationally unattractive. An alternative approach for handling moving boundary conditions is the small-perturbation techniques [9–12]. Gao et al. [9] reported a small-perturbation boundary condition method for the Euler equations on non-moving Cartesian grids. The essential idea is to use the classical small-disturbance potential flow method, to approximate the solid wall boundary conditions for an airfoil by a first-order expansion on the airfoil mean line. Computations of both steady and unsteady problems involving even relatively thick airfoils and moderately large angles of attack show excellent accuracy for an unsteady problem. Strictly the method is limited to thin airfoil assumption and treatment of stores and fuselage is neither straightforward nor realistic with the method. Yang et al. [10,11] used stationary body-conforming grids, which did not require the assumption of thin geometry. Small-perturbation boundary conditions were not used to mimic the geometry. Instead they were used only to track the movements of two and three-dimensional surfaces such as the airfoil or wings. Kirshman et al. [12] also developed similar approach but on a Cartesian grids in their studies of two-dimensional airfoils. As applied to flutter simulations where the unsteady motion or deformation of the flow boundary is small with respect to the mean position, the small-perturbation approximation proves to be general and accurate. In Kirshman and Liu's two-dimensional Cartesian grid method [12], they made use of a set of shape functions over a cloud of gridless nodal points and incorporated the unsteady perturbation of the boundary conditions by a perturbation of the shape functions.

The aim of the present work is to address the problems in using Cartesian grids for computing complex static or moving geometries in three dimensions. A different approach is adopted in the present three-dimensional studies. The approach presented here makes use of a Cartesian grid for handling complex geometries and small-perturbation assumption to avoid the need for moving grid algorithm. Though possible, the direct extension of the Cartesian grid solvers with the gridless boundary condition of Kirshman et al. [12] and Koh et al. [5] for three-dimensional problems is not efficient. The present method of boundary implementation does not require the solution of the flow equation for boundary condition implementation and is direct and considerably simpler especially when applied in three dimensions. The approach makes use of reflected points or ghost cells. A gridless cloud of points is needed to obtain the variables for the reflected node via a least-squares approximation. It is found to be more efficient especially when used for three-dimensional problems. Euler fluxes for the neighbors of cut cells are directly computed using the reflected points which are set to satisfy the moving and non-moving wall boundary conditions. A finite-volume formulation with central differencing for the Euler equations is used throughout the rest of the flow field [13,14]. As will be discussed below to implement the small-perturbation method, the flow variables at the reflected points are selected such that the perturbation in the surface normal of the moving surface is taken into account. The inherent simplicity, accuracy, robustness, and computational cost of the proposed solution procedure, makes this an attractive approach for the envisioned application.

2. Numerical algorithms

2.1. Governing equations

The governing equation for the three-dimensional unsteady Euler equations in integral form is given as follows,

$$\frac{\partial}{\partial t}\int_V \vec{U}\,dv + \int_S \vec{F}\cdot n\,d\vec{S} = 0, \tag{1}$$

where V denotes a control volume with closed boundary surface S, and $\vec{n}$ is the outward normal vector on S. The dependent variable $\vec{U}$ and the flux vector $\vec{F} = f_x\vec{e}_x + f_y\vec{e}_y + f_z\vec{e}_z$ are given by,

$$U = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{Bmatrix}, \quad f_x = \begin{Bmatrix} \rho(u-u_b) \\ \rho u(u-u_b)+P \\ \rho v(u-u_b) \\ \rho w(u-u_b) \\ \rho H(u-u_b) \end{Bmatrix}, \quad f_y = \begin{Bmatrix} \rho(v-v_b) \\ \rho u(v-v_b) \\ \rho v(v-v_b)+P \\ \rho w(v-v_b) \\ \rho H(v-v_b) \end{Bmatrix}, \quad f_z = \begin{Bmatrix} \rho(w-w_b) \\ \rho u(w-w_b) \\ \rho v(w-w_b) \\ \rho w(w-w_b)+P \\ \rho H(w-w_b) \end{Bmatrix}, \tag{2}$$

where P, ρ, E and H denote the pressure, density, total energy and total enthalpy, respectively, while (u, v, w) are the Cartesian velocity components and (u_b, v_b, w_b) are the moving velocity of the grid points. The unit vectors along the Cartesian directions are defined by e_x, e_y and e_z, respectively. For a perfect gas specific heat ratio γ, the total energy is given by,

$$E = \frac{P}{(\gamma-1)\rho} + \frac{1}{2}(u^2 + v^2 + w^2), \tag{3}$$

and the total enthalpy by,

$$H = E + \frac{P}{\rho}. \tag{4}$$

This expression represents a system of equations corresponding to conservation of mass, momentum and energy.

2.2. Spatial and temporal discretization

Spatial discretization of the Euler equations for all the fluid nodes in the Cartesian grid is performed using the standard finite-volume method. The governing equations are solved using a cell-centered finite-volume scheme. Applying Eq. (1) to each cell in the mesh, a set of ordinary differential equations are obtained as follows,

$$\frac{d}{dt}(U_{i,j,k}V_{i,j,k}) + R(U_{i,j,k}) = 0, \tag{5}$$

where the residual $R(U_{i,j,k})$ is defined as,

$$R(U_{i,j,k}) = Q_{i,j,k} + D_{i,j,k}. \tag{6}$$

Here, $Q_{i,j,k}$ is the net flux out of the cell and $D_{i,j,k}$ contains the artificial dissipative terms. The artificial dissipative terms are made up of a blend of second-order and fourth-order differences to provide first-order dissipation around shocks and third-order dissipation in smooth flow regions [13].

For Cartesian grids, boundary cells near the object surfaces have one or more neighbors that are cut by the object. These cells have an incomplete stencil for flux computations. The missing values needed by the boundary cells for flux computation will be provided via the boundary condition implementation.

In this study, the dual-time method proposed by Jameson [15] is used to perform time-accurate calculations. The basic implementation of the dual-time-step method is the same as in Ref. [16]. A second-order accurate fully implicit scheme is then used to discretize the derivative of the physical time. The Runge–Kutta multi-stage time integration [13] is applied to march the solution to convergence in pseudo-time. The modified multi-stage Runge–Kutta scheme is used as it gives the best ratio of allowable time step to computational work per time step. The integration is given as follows,

$$\begin{aligned}
U^{(0)}_{i,j,k} &= (U^{*}_{i,j,k})^{m},\\
U^{(1)}_{i,j,k} &= U^{(0)}_{i,j,k} - \frac{1}{4}\frac{\Delta t^{*}}{V_{i,j,k}}R^{*(0)},\\
U^{(2)}_{i,j,k} &= U^{(0)}_{i,j,k} - \frac{1}{3}\frac{\Delta t^{*}}{V_{i,j,k}}R^{*(1)},\\
U^{(3)}_{i,j,k} &= U^{(0)}_{i,j,k} - \frac{1}{2}\frac{\Delta t^{*}}{V_{i,j,k}}R^{*(2)},\\
U^{(4)}_{i,j,k} &= U^{(0)}_{i,j,k} - \frac{\Delta t^{*}}{V_{i,j,k}}R^{*(3)},\\
(U^{*}_{i,j,k})^{m+1} &= U^{(4)}_{i,j,k}.
\end{aligned} \tag{7}$$

The scheme is explicit with a Courant–Friedrichs–Lewy (CFL) condition of CFL $\leqslant 2\sqrt{2}$ and has second-order accuracy in time for a non-linear equation. For steady-state solutions, local time stepping is used to accelerate convergence. With this, the time step for each cell varies and is based on applying the CFL criterion to the local flow condition.

2.3. Embedded multigrid strategy

The efficiency of the Cartesian flow solver can be greatly enhanced with a multigrid implementation. By exploiting the use of successively coarser sets of grids where each grid level is responsible for removing a particular bandwidth of errors, the solution is accelerated by time stepping via a combinational sequence of fine and coarse grids. Here in this paper, a multigrid with mesh embedding for local refinements is adopted. The main aim of using embedded meshes for the multigrid process is the great reduction in the number of computational cells used as compared to traditional multigrid process. Finer meshes are embedded only at regions where flow and geometry refinement are needed. The grids in one level have to be fully embedded in the previous coarser level to ensure proper nesting of grids. This is important for the multigrid process where transfer of information takes place from one grid to another. This approach provides a means of strongly-coupled communication in the hierarchy of Cartesian grids.

In the traditional multigrid approach, the finest computational grid spans the entire domain and successively coarser grids are formed by omitting every other grid point of the finer grid. The multigrid method used here is somewhat different. It is based on the concept of a hierarchy of embedded grids. The embedded multigrid is used to couple the different grid blocks and accelerate the computation in the Cartesian grid system. A large Cartesian grid work as a background grid within which all the other grid blocks could communicate. The finer grids are responsible for near body solution while the coarser grids are responsible for dealing with the other part of the flow field in the computational domain.

Grid levels with different resolution are generated prior to flow field evolution. All the grids in each level have to be fully embedded in the previous level of coarser grids. Every coarse grid with finer grids embedded in it can be classified into two regions: overlapping and non-overlapping. A finer grid is embedded in the overlapping region and the corresponding overlapping coarse grid provides the correction for the finer grid in multigrid process. The transfer of information for the multigrid process involves only nodes in the overlapping regions. These nodes accept an effective forcing function from the finer mesh points. The forcing functions for nodes in the non-overlapping regions are set to zero. In this way, the multigrid routine can be conveniently applied to the entire grid but is only effective in the overlapping regions. The outer boundary of an embedded finer mesh is specified with values from the coarser mesh based on a Dirichlet condition.

The multigrid method discussed above centers around the Full Approximation Storage (FAS) technique, where multigrid is applied directly to the non-linear form of the governing equations. The method uses a grid embedding technique with successively finer grid for improved accuracy in the solution. The main advantage of this is that it helps greatly in reducing the amount of grid cells required to achieve the same flow resolution as compared to traditional multigrid methods. The strategy for the information transfer between coarse and fine grids in the overlapping region is the same as that used in Ref. [17]. For maximum effectiveness, a multigrid sequencing strategy will also be used. Computation begins with the coarsest level and advances progressively to the finest level using the multigrid method at the intermediate coarse grid as shown in Fig. 1. The solutions on the coarse grids also provide good initial solutions on the subsequent finer grids.

In the current study, all grid levels with different resolution are generated prior to the flow field computation. The grid refinement is used to ensure adequate resolution in the specific areas, especially near the shock and curving wall boundary. For example, in order to resolve the curvature near the wing leading edge, the cell dimensions of the finest Cartesian grid are made to be approximately equal to or less than the surface-grid dimensions. At the present, this process is still based on ad-hoc criteria by the user. An automated grid adaptation process will be pursued in future work. In this work, homogeneous refinement is used, which means the Cartesian grid is refined locally in all three dimensions.

2.4. Boundary conditions

No special treatment is used for far field boundary conditions. The usual characteristic analysis based on one-dimensional Riemann invariants is used to determine the values of the inviscid flow variables for the far field of outermost block. As with the boundary conditions on the surface in the Cartesian method, it is important to first classify the nodes near the object boundary as seen in Fig. 2. Four types of nodes are identified. Type 1 nodes are cells whose Euler fluxes can be computed easily in the finite-volume method since the values of all its neighbors are defined, i.e., the computational stencil for the finite-volume method is complete in the second-order scheme. These cells are termed *fluid nodes*. Type 2 nodes are cells that fall inside the object. These include cells that are partially cut by the object surfaces but with its cell-center lying inside the object. These cells are termed *solid nodes*. Type 3 nodes are cells that have solid nodes as neighbors and they are termed *boundary nodes*. Type 4 nodes are intersection points on the surfaces where boundary conditions will be imposed. These are termed *surface nodes*.

Computational nodes are made up of both the fluid nodes and the boundary nodes. Unlike the fluid nodes, the Euler fluxes of the boundary nodes cannot be easily computed as the stencil for the finite-volume computation is incomplete. The missing solid node values needed to complete the stencil will be provided via the implementation of surface boundary conditions, which makes use of reflected points which are determined via a gridless method. In this way, the flux computation for both the fluid nodes and boundary nodes can be computed in the same manner.

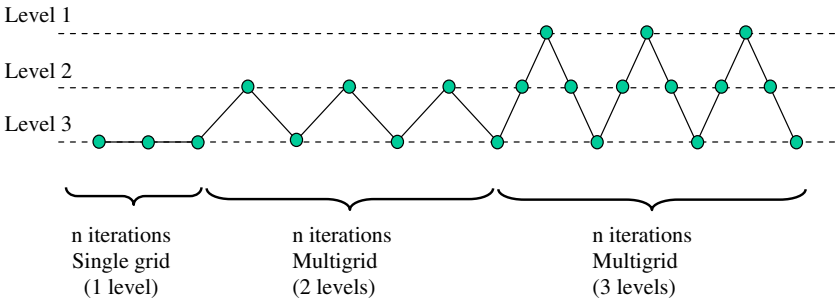


Fig. 1. Multigrid sequencing strategy.

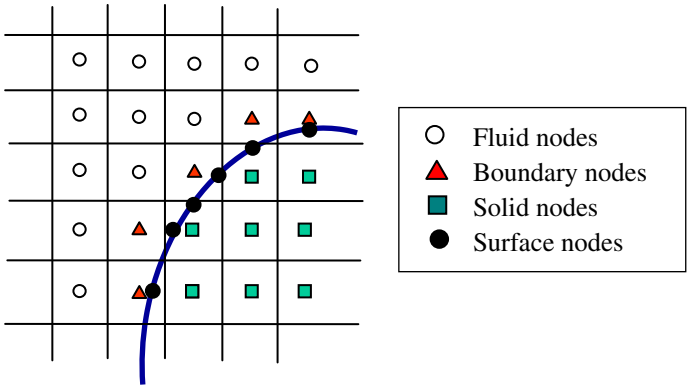


Fig. 2. Classification of nodes for surface boundary conditions.

2.4.1. Surface boundary conditions

For inviscid flow, a slip condition is used. The boundary implementations involve imposing the boundary conditions at the surface nodes as well as providing the missing solid node values needed to complete the flux computations. An illustration of the implementation process is shown in Fig. 3. The line joining the boundary node and the solid node intersect the surface boundary at the surface node. From the surface node, the normal vector to the surface is defined and a tangent plane is constructed. The solid node is reflected about this tangent plane to obtain a reflected node. Since the reflected node is nested in the flow field, its values are found by interpolating from the neighbouring computational nodes. It is important to include the slip condition at the surface node when interpolating for the reflected node values. Thus, the normal velocity component of the reflected node is interpolated with the additional information that the normal component of velocity is zero at the surface node. The reflected node values are then used to specify the values at the solid node such that the boundary conditions are satisfied across the surface boundary.

The value of each of the variables at the solid node is set as follows,

$$\begin{aligned}
(V_n)_S &= -(V_n)_R,\\
(V_{t_1})_S &= (V_{t_1})_R,\\
(V_{t_2})_S &= (V_{t_2})_R,\\
(\rho)_S &= (\rho)_R,\\
(P)_S &= (P)_R + d\frac{\partial P}{d\vec{n}},
\end{aligned} \tag{8}$$

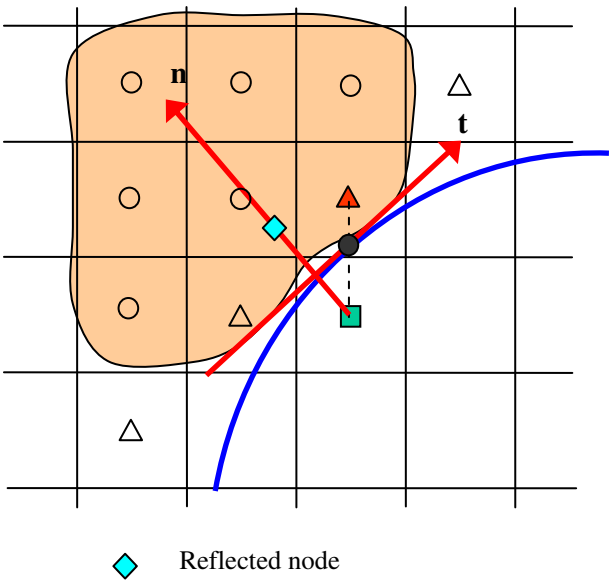


Fig. 3. Implementation of surface boundary conditions.

where d is the distance between the reflected node R and the solid node S. The subscript $\vec{n}$ stands for the normal direction of the boundary surface; $\vec{t}_1$ and $\vec{t}_2$ are two orthogonal directions on the tangential plane. The normal pressure gradient is given as follows,

$$\left(\frac{\partial P}{\partial \vec{n}}\right)_p = -\frac{\rho V_t^2}{K}, \tag{9}$$

where K is the local curvature of the surface in the direction of the flow. For a three-dimensional surface, the curvature in the direction of the flow on the surface must be used. Taking this curvature of the body into consideration will result in the improvement of the accuracy. The location of the shock on the surface is more accurate and spurious entropy productions are reduced with this correction for the pressure boundary condition [18].

This method of boundary implementation is direct and simple. In addition, the values of the missing solid nodes needed for Euler fluxes computation can be provided at the same time. The information pertaining to the surface nodes are associated with the boundary nodes and are provided in the preprocessing stage prior to flow computation. A boundary node can have more than one associated surface nodes depending on the number of intersections it has with the object surface. The boundary nodes are managed using an unstructured data approach. As such, the boundary implementation is performed in a separate routine which makes it flexible and can be easily implemented into other schemes.

2.4.2. Interpolation using least-squares approximation

To implement the boundary conditions the variables of the reflected nodes are approximated from the computational nodes in its vicinity. For each reflected node, a cloud of N gridless points around it is selected. With a group of N cloud points in three dimensions, the least-squares approximation of a function $f(x,y,z)$ can be generally written as

$$f(x,y,z) = a_0 + a_1 x + a_2 y + a_3 z + a_4 xy + a_5 xz + a_6 yz + \cdots + a_M x^i y^j z^k, \tag{10}$$

which consists of $M+1$ polynomial terms where $a_0, a_1, \ldots, a_M$ are the $M+1$ coefficients for the least-squares approximation and i, j, k are the highest order of x, y, z. The least-square approximation given by Eq. (10) can be truncated to a given order

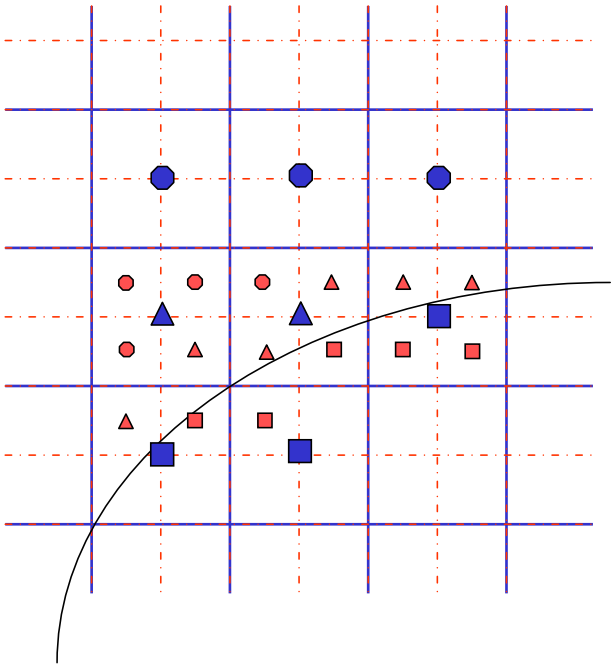


Fig. 4. Multigrid implementation at surface boundary.

Fig. 5. Description of the movement of the airfoil.

based on the accuracy requirement. Generally, a higher-order approximation will encounter increased computational effort because of the needed increase of cloud points and also possible numerical instability. To obtain stable solutions with good accuracy while minimizing the computational cost, the following bi-linear interpolation is used in the present study,

$$f(x,y,z) = a_0 + a_1x + a_2y + a_3z + a_4xy + a_5xz + a_6yz. \tag{11}$$

Writing out the system of equations with the N selected points gives,

$$\{F\} = [A]\{a\}, \tag{12}$$

where

$$\{F\} = \begin{Bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{Bmatrix}, \quad \{a\} = \begin{Bmatrix} a_0 \\ a_1 \\ \vdots \\ a_6 \end{Bmatrix}, \quad \{A\} = \begin{Bmatrix} 1 & x_1 & y_1 & z_1 & z_1 & x_1y_1 & x_1z_1 & y_1z_1 \\ 1 & x_2 & y_2 & z_2 & z_2 & x_2y_2 & x_2z_2 & y_2z_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & z_N & z_N & x_Ny_N & x_Nz_N & y_Nz_N \end{Bmatrix}. \tag{13}$$

Performing a least-squares curve fit on each variable respectively, the following is solved to obtain the set of coefficients {a}:

$$\{a\} = ([A]^T[A])^{-1}[A]^T\{F\}. \tag{14}$$

As shown in Eq. (13), when a least-squares curve fit is applied to approximate the variables on a reflected node, N cloud points are chosen to determine its coefficients. The selection of the cloud points depends on the following two criteria. First, they must be fluid nodes that are as close as possible to the reflected node. Second, they should be located on the outward normal direction of the wall to avoid including points from the wrong side as in case of thin geometry. Different numbers of cloud points may be used for different cases. In the current study, at least 7 cloud points are used to determine the coefficients of the least-squares approximation given by Eq. (11). With the respective sets of coefficients found, the variables of the reflected node can easily be obtained using Eq. (11).

2.4.3. Multigrid implementation at surface boundary

As mentioned above, the multigrid with embedded meshes for local refinement has been implemented in the present study. Careful consideration should be taken for the treatment of boundary conditions on the cutting surface of sold walls. The strategy used in the current study is shown in Fig. 4. During the restriction step of multigrid, the values of the coarse grid points are interpolated from surrounding fine grid values. The values at the boundary *solid nodes* are provided via surface boundary condition implementation. Residuals of coarse grid are obtained by summing up surrounding fine grid residuals. Residuals of boundary *solid nodes* are set to zero. During the prolongation, correction on a coarse grid is transferred to a finer

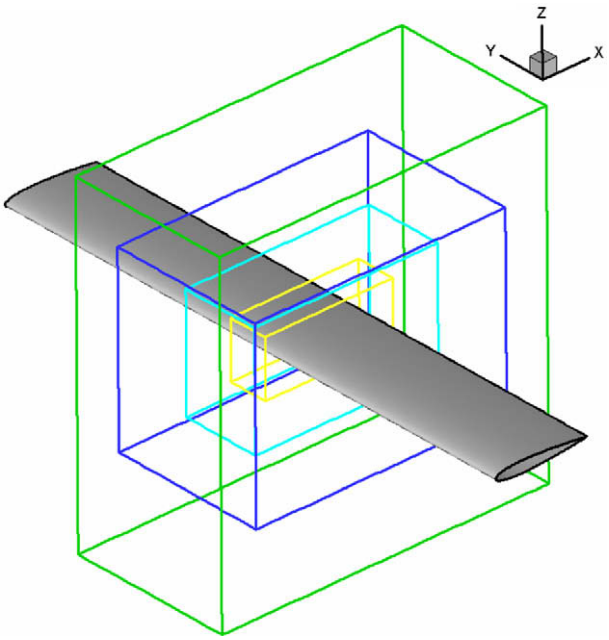


Fig. 6. 3D view of embedded Cartesian grids for a uniform NACA0012 wing.

grid. The values of the boundary *solid nodes* on coarse grids are required and provided via boundary condition implementation.

2.4.4. Small-perturbation method

The use of moving or deforming grid can be a very time-consuming and nontrivial task in an unsteady computation for practical applications. In the current study, a small-perturbation boundary treatment is utilized to avoid the tedious process of grid regenerations as proposed by Refs. [9–12]. This treatment makes it possible to use the stationary Cartesian grids for unsteady flow computations with moving boundaries. This is also applicable for flutter simulations due to that the displacement of the surface from the original position is usually small in an aeroelastic problem. As shown in Fig. 5, in an unsteady computation, considering a new point $P(x, y, z)$ just moving from its previous position $P_0(x_0, y_0, z_0)$ on the original surface with a small displacement, we can have the following velocity boundary condition at P,

$$V \cdot n = V_b \cdot n, \tag{15}$$

where V_b is the moving velocity of the surface and n is the unit normal vector at the new position P. With the small perturbation method, a Taylor series expansion is applied here to approximate the variables at the point P with those at P_0. Thus, Eq. (15) can be written as follows by taking a first-order approximation,

$$V(x_0, y_0, z_0) \cdot n = V_b \cdot n. \tag{16}$$

If needed, a higher-order approximation can be achieved by utilizing the Taylor series expansion approach. As shown in the above equation, although the flow velocities are evaluated at the stationary position P_0, the normal vector n is computed at the actual point P, which changes with time. In addition, the moving velocity V_b is also time-dependent in an unsteady motion. The small perturbation method can be easily incorporated into the boundary treatment presented earlier by a simple

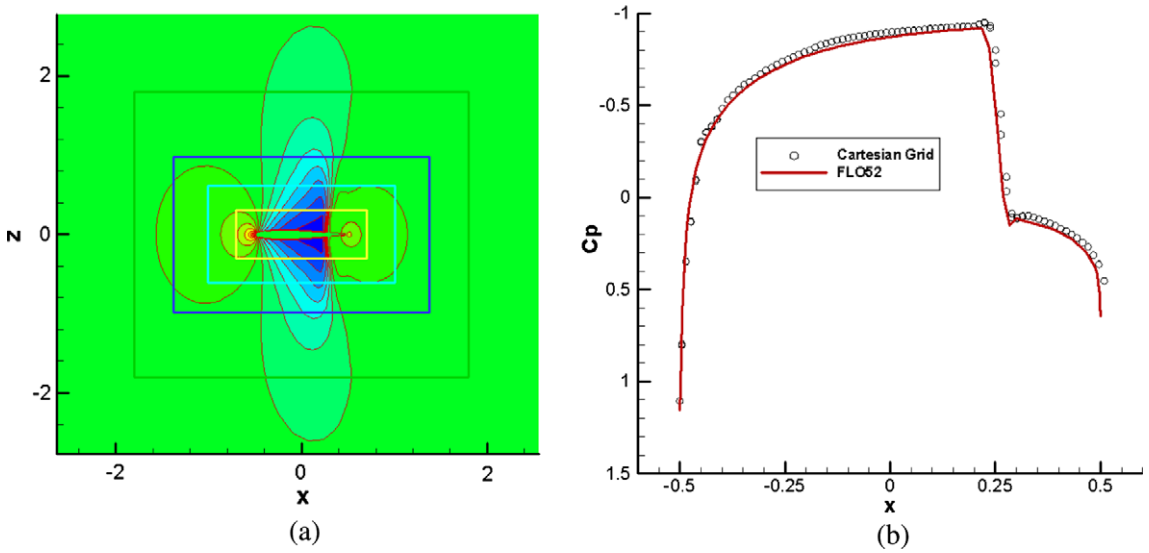


Fig. 7. Pressure plots around a uniform NACA0012 wing: (a) Cp contours; (b) Cp distribution.

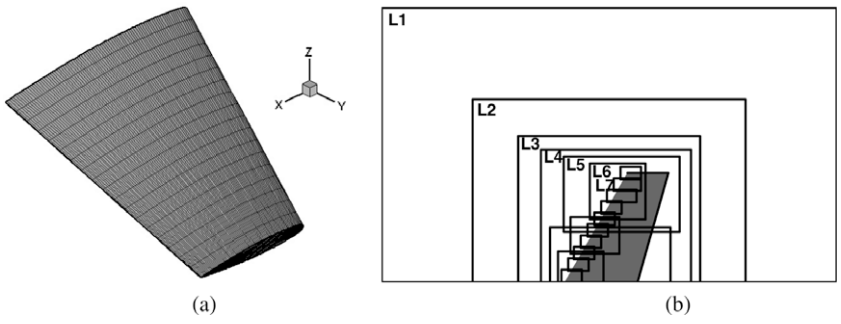


Fig. 8. Grid for the ONERA M6 wing: (a) surface grid; (b) embedded grid structure.

and direct way. For unsteady computations, the boundary condition imposed on the surface nodes becomes $V \cdot n = V_b \cdot n$ instead of imposing the condition of $V \cdot n = 0$ on the surface nodes as discussed previously.

3. Results and discussion

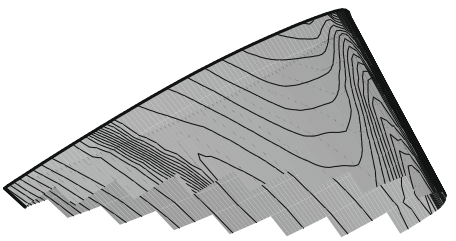
To validate the accuracy and efficiency of the 3D Cartesian Euler solver using the method of gridless boundary implementation, five test cases are presented in this section. The first three test examples are the steady flows over a uniform NACA0012 wing, ONERA M6 wing and the RAE wing-body configurations, respectively, to demonstrate the accuracy and the robustness of the boundary condition method used in the Cartesian solver before attempting time-accurate simulations using the perturbation boundary condition. The last two unsteady cases, including a pitching case for a NACA0012 and a flutter case for the AGARD 445.6 wing, are presented to demonstrate the suitability of the small perturbation boundary condition method in conjunction with the gridless method to account for the moving surfaces. We apply the coupled CFD–CSD method [19] to simulate the aeroelastic system directly in the time domain, and to determine aeroelastic stability based on the computed time-histories of generalized displacement at the flight conditions of interest. It is shown here that the current methods are accurate and robust in handling transient effects.

3.1. Steady flow over a uniform NACA 0012 wing

In this case, we compute a relatively simple case, 3-D flow over NACA 0012 wing at Mach number $M_\infty = 0.85$ and angle of attack $\alpha = 0^\circ$, to validate accuracy of the current Cartesian Euler solver using the embedded multigrid strategy with the method of gridless boundary implementation. Fig. 6 shows the three-dimensional view of embedded Cartesian grids for a uniform NACA0012 wing. It can be seen that the computational grid used here includes 4 levels of mesh embedment. We obtained the steady-state solution for this transonic flow based on the embedded Cartesian grids. The pressure contour plot on the wing surface is shown in Fig. 7(a). The smooth transition for the pressure contours can be observed among the different grid levels. Fig. 7(b) shows the comparison of the Cp distributions produced by the current code and a finite-volume code using body-fitted grids, FLO52 developed by Jameson [13]. Excellent agreement has been observed between them.

3.2. Steady flow over the ONERA M6 wing

The further investigation for the inviscid flow over the ONERA M6 wing, which is a real three-dimensional case, has been done here to demonstrate the effectiveness of the boundary condition treatment for steady flow simulations. The ONERA M6 wing is a classic CFD validation test used in numerous papers to validate CFD codes. The current computation was performed at the standard test conditions of Mach 0.84 and angle of attack, $\alpha = 3.06^\circ$. We note here that although the effects of angle of attack can be included and imposed by the perturbation boundary condition, we do not use it here. Instead the angle of attack is imposed as normally done via the inflow boundary conditions. The computational grid, having about two millions of grid points totally, consists of 7 levels of mesh embedment with a total of 19 grid blocks, in which L1 is the coarsest grid level while L7 represents the finest level. Fig. 8(a) shows the surface mesh on the wing and Fig. 8(b) shows the arrangement of the grid blocks used to compute the flow. Grid embedment allows for finer grid blocks to be placed at regions that require



geometry or flow refinement. Hence finer grid blocks are located close to the object surface and at specific places including the leading or trailing edges of the wing.

The steady-state solution for the transonic flow is obtained and the pressure contour plot on the wing surface is shown in Fig. 9. Smooth contours are observed in the plot. Fig. 10 compares the computed pressure distributions for this wing with experimental data by Schmitt and Charpin [20] at four different locations along the span (20%, 44%, 65%, and 90% span). The results show that at the 20%, 44% and 65% stations, the upper surface of the wing exhibits two relatively weak shock waves along the chord. At these spanwise stations, the forward shock waves are well captured. The second shock wave is predicted slightly down stream of the experimental shock locations. This slight discrepancy is typical of inviscid flow computations reported by other authors and can be attributed to the viscous effects which the flow model of Euler method does not include. As the spanwise distance increases, the two weak shocks move closer to each other and eventually merge to form one, which is a relatively strong shock wave. At the 90% station, the single shock wave is predicted with a very good agreement with experimental results. Overall, the present results match well with the experimental results. The general trend of pressure distributions along the wing surface, the leading edge suction peak, the magnitudes and locations of the computed shock waves are in reasonable accordance with the experimental results.

Fig. 11 shows the convergence history in terms of density residual for this case on all the mesh levels. Our multigrid strategy is coupled with the multi-layer grid embedment process. The multigrid levels used in the multigrid acceleration method are the levels of overlapping grid embedment. The grid embedment is classified into two types: overlapping and non-over-

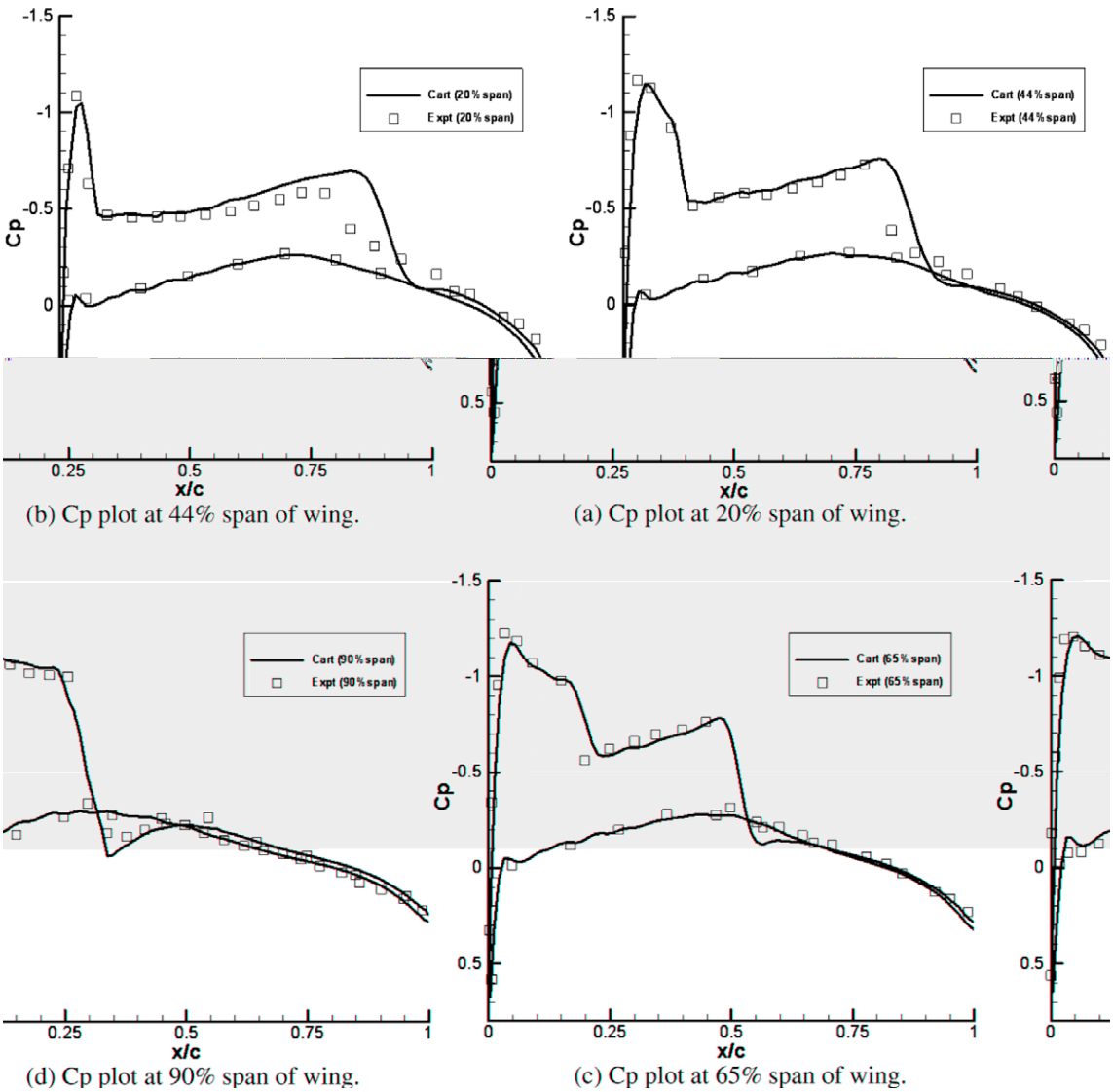


Fig. 10. Cp distribution on the surface of the ONERA M6 wing at Mach = 0.84, $\alpha = 3.06^{\circ}$.

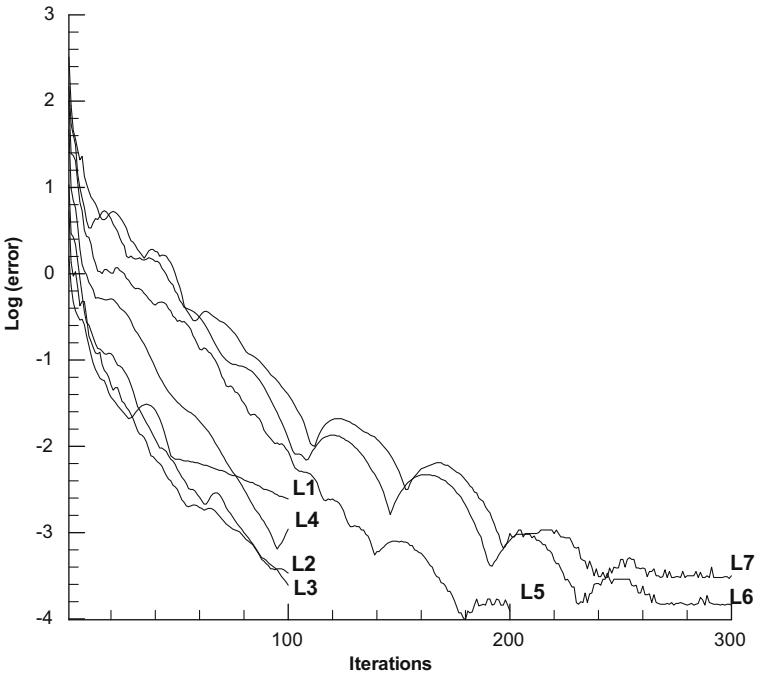


Fig. 11. Convergence plot for the ONERA M6 wing at Mach = 0.84, $\alpha = 3.06^{\circ}$.

lapping. A finer grid may be embedded within a coarser grid. This fine grid is actually overlaid over the coarse grid. Several such non-overlapping fine grids may be embedded in the same background coarse grid, forming two levels of multigrid: the coarse background grid and the fine grid that consists of the patches of non-overlapping finer grids connected by the coarse grid. Multiple layers of such embedded grids can be formed recursively to improve local grid resolution and at the same time forming the multigrid levels for convergence acceleration purpose. As such, we notice that the L1 grid consists of the single coarse background grid; the L2 grid has one layer of embedment (finer grid) and thus also one extra multigrid level that can be used for convergence acceleration; similarly, the L(N + 1) grid has one more level of multigrid than the L(N) grid, but the finest level grid in the L(N + 1) grid is one level finer than that of the L(N) grid. Therefore, it is possible that the convergence on the L(N + 1) grid is slower than that on the L(N) grid despite the extra level of multigrid simply because the L(N + 1) has more grid points in its finest grid than those on the L(N) grid. This explains why the convergence rates on the L5, L6 and L7 could be slower than those on the L1–L4 grids. Ideally we would like to achieve the same convergence rates independent of the grid refinement/embedment levels because we at the same time increase the multigrid acceleration levels. This seems to be the case as we go to the finer levels of refinement on the L5, L6 and L7 grids. It can be seen that the maximum residual reduces by more than 5 orders of magnitude on the finest grids in about 300 iterations.

3.3. Steady flow over RAE wing-body configuration

One of the main advantages of using Cartesian grid is in handling complex configurations in three-dimensional cases. In the current Cartesian solver, a complex geometry is broken down into smaller components and treated individually on a component basis in the preprocessor stage. This makes it easier for the preprocessor to find the intersection points of the Cartesian grid blocks with each of the surface grids and store the geometrical properties needed for the flow computation. This information pertaining to the intersection points is stored in an unstructured manner. Hence, the flow computation is regardless of the number of components that form the object. For this problem, the wing and the fuselage are two separate components and their composite is shown in Fig. 12(a). The wing is finely defined and is allowed to intersect into the fuselage. No special effort is needed to determine the intersection as it suffices just to determine the flow grids among the Cartesian grids.

To demonstrate the capability of the current Cartesian code in handling multiple components, the Cartesian solver is employed to compute the three-dimensional flow over a transport wing-fuselage configuration as seen in Fig. 12(a). The flow is computed at Mach number 0.9 and angle of attack, $\alpha = 1.0^{\circ}$. The grid structure for the RAE wing-body case is set up as shown in Fig. 12(b). The computational grid consists of a total of 16 blocks through 7 levels from the coarsest level L1 to the finest level L7, containing a total of near four million grid points. The finest 2 levels of grids L6 and L7 are for the refinement of flow on the wing as seen in Fig. 12(c).

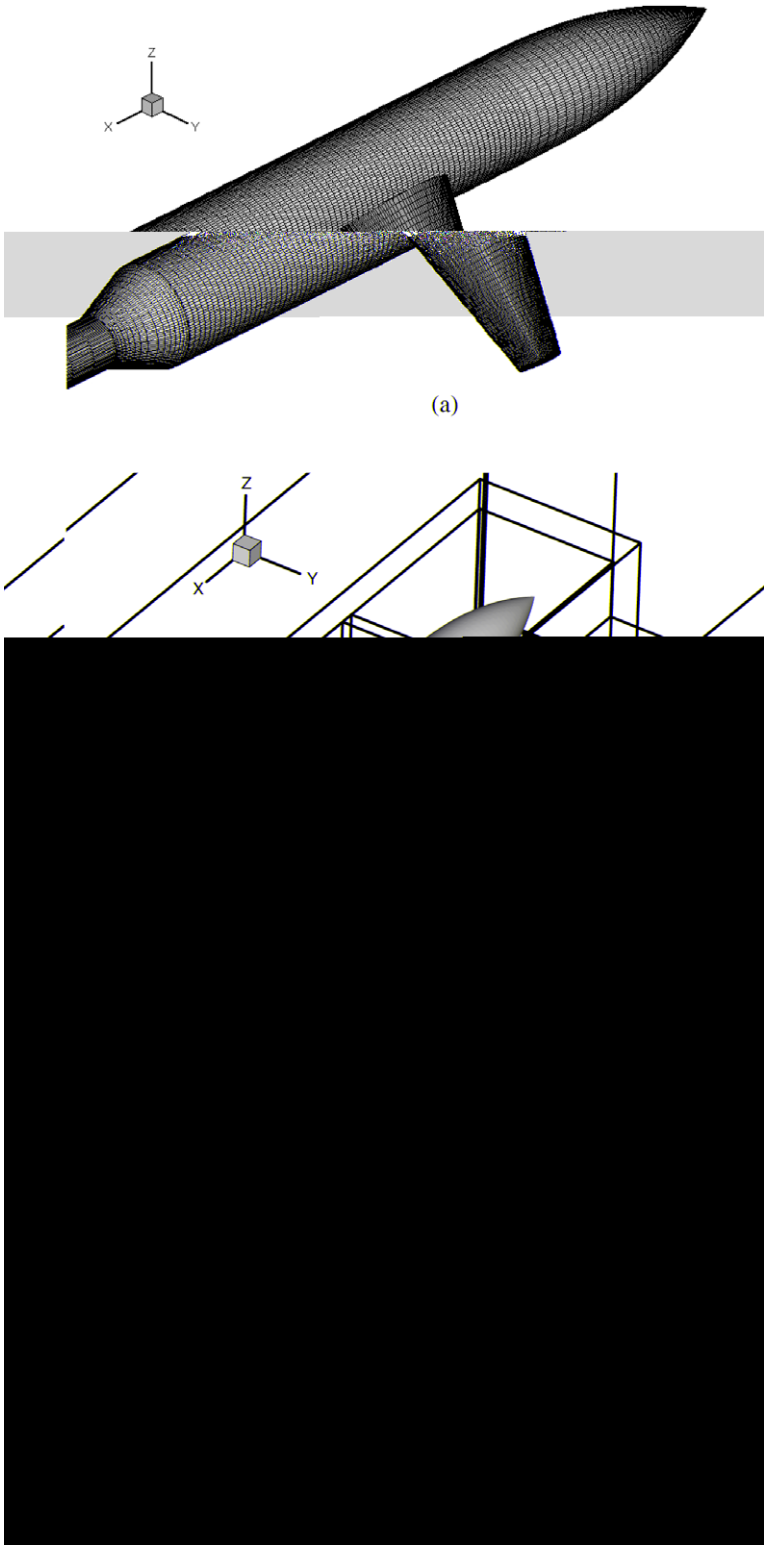


Fig. 12. Grid structure for the RAE wing-body configurations: (a) surface grid; (b) grid blocks; (c) embedded grid structure.

The transonic solution is computed and the pressure contours on the surface of the wing and body are shown in Fig. 13. To check the accuracy of the solution on the surface of the wing, the Cp distribution along the span of the wing is also plotted

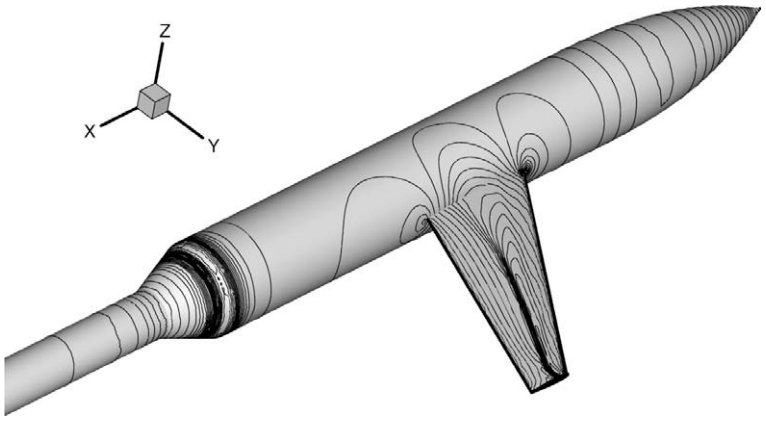


Fig. 13. Pressure contour plot on the surface of the wing-body at Mach = 0.9, $\alpha = 1.0^{\circ}$.

against experimental results [21]. The Cp plot is obtained at 25%, 60% and 92% of the span, respectively. As seen from the plots in Fig. 14, the solutions obtained by the present method are in very good agreement with the experimental results. Fig. 15 illustrates the convergence history for the RAE wing-body case on all the seven mesh levels. Fast convergence can be observed again because of the embedded multigrid strategy. The maximum residual reduces by around 6 orders of magnitude in less than 200 iterations on the finest grids.

3.4. Unsteady flow over a pitching airfoil – AGARD-CT1

The present method is used to calculate an unsteady flow over a pitching NACA0012 at a free-stream Mach number of 0.6. The pitching motion of the airfoil is described by

$$\alpha(t) = \alpha_m + \alpha_0 \sin \omega t$$

where α_m, α_0 are the mean angle of attack and the amplitude of the oscillation, respectively. For this analysis, the mean angle of attack α_m is prescribed by the far field boundary condition, and perturbation of the surface normal will be used to account for the oscillating pitch of the airfoil. The angular frequency ω is related to the reduced frequency defined by

$$\kappa = \frac{\omega c}{2U_\infty}$$

In this case, we choose $\kappa = 0.0808, \alpha_m = 2.89^{\circ}, \alpha_0 = 2.41^{\circ}$.

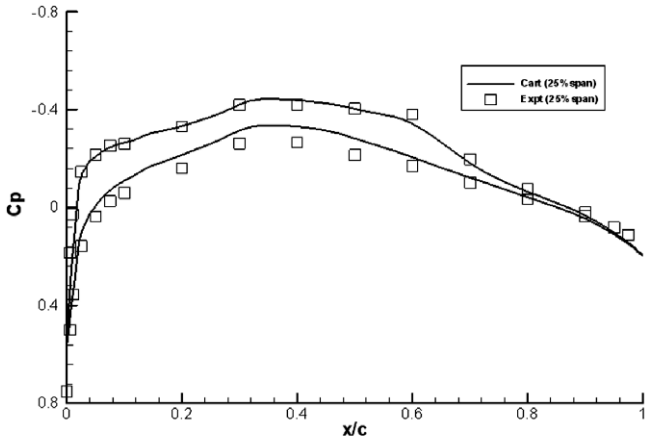
The computational mesh used for this simulation is identical to that used for the steady-state simulation as shown in Fig. 6. The unsteady simulation is initiated from a steady flow solution with the mean angle of attack prescribed as a far field condition. Fig. 16 compares the current predictions with the experimental data referred to the AGARD-CT1 test case [22] by means of the variation in lift coefficient versus angle of attack. Reasonable agreement can be found between them.

3.5. Unsteady flow over AGARD 445.6 wing

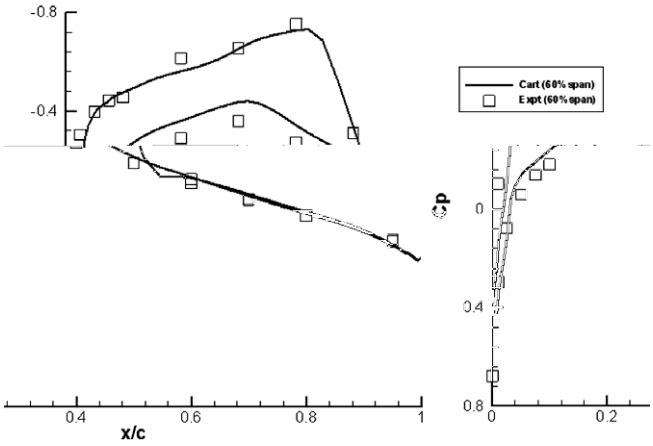
In this case, we compute the aeroelastic behavior of the AGARD 445.6 wing at a range of Mach numbers in the transonic regime, and map out the flutter boundary. Computations are performed to assess the reliability and accuracy of the current methods for aeroelastic simulation. The present computational results for the AGARD 445.6 wing are compared with published numerical and experimental results. The AGARD 445.6 wing has a quarter-chord sweep angle of 45° and its cross-section is given by the NACA65A004 airfoil. The flutter characteristics of this wing were investigated experimentally over a wide range of Mach numbers [23]. These results were documented as an AGARD standard aeroelastic configuration [24] and have been widely used to test and validate flutter calculations since then.

The reduced frequency κ is introduced as a measure for the unsteadiness of the flow and calculated by $\kappa = 2\omega b/U_\infty$ in which U_∞ is the free-stream velocity, ω is the characteristic oscillating frequency, and b denotes the half-chord at wing-root. The speed index v_f is defined by $v_f = U_\infty/(b\omega\sqrt{\mu})$. Here the mass ratio μ represents a non-dimensional wing mass given by $\mu = m/(\pi\rho_\infty b^2)$, where m is the wing mass per unit span and ρ_∞ is the free-stream air density. The simulations from both the body-fitted grid code and the present Cartesian solver are conducted using an implicit time step $\Delta t = T/32$, where T is the time period based on a reference frequency. In this work, the first torsion modal frequency with the value of 38.17 Hz is taken as the reference frequency as used in Ref. [24].

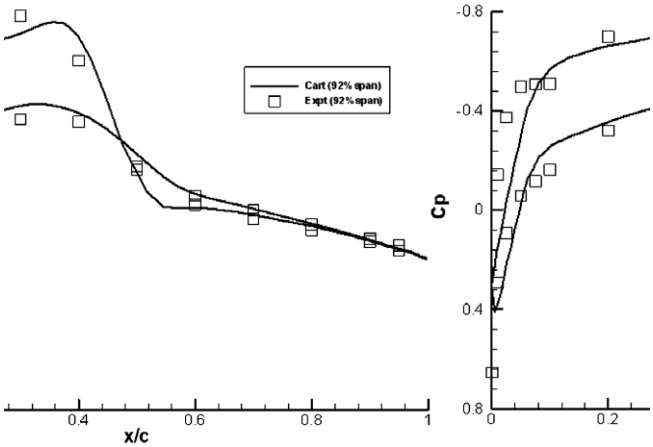
In this set of computations, we consider the weakened wing model as listed in Ref. [23]. The fluid grid and structure grid of the wing are depicted in Fig. 17(a). The structure of embedded Cartesian grids used is shown in Fig. 17(b). The Cartesian



(a) Cp plot at 25% span of the wing.



plot at 60% span of the wing. (b) Cp



plot at 92% span of the wing. (c) Cp

Fig. 14. Cp distribution on the surface of the wing at Mach = 0.9, $\alpha = 1.0^{\circ}$.

grid used has a total of 13 blocks embedded to 7 levels with a total of about one million grid points. The wing is modeled by the first five natural vibrational modes identified as the first bending, first torsion, second bending, second torsion and third

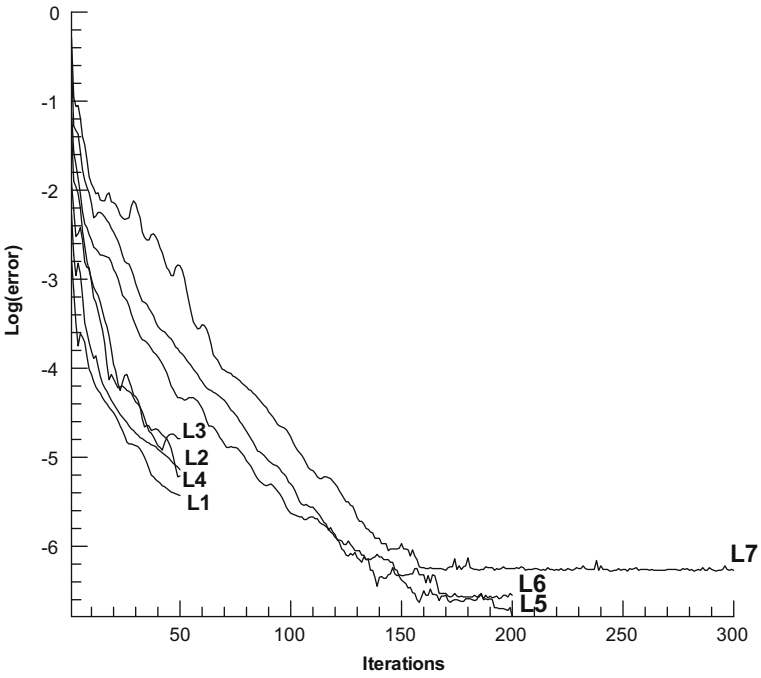


Fig. 15. Convergence plot for the RAE wing-body.

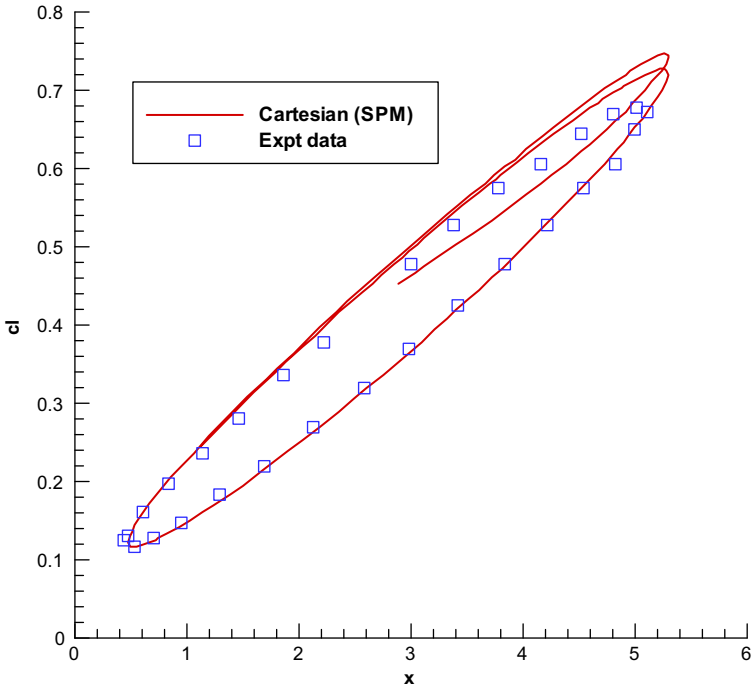


Fig. 16. Comparison of computed lift coefficient with experimental data for prescribed oscillation of NACA 0012 airfoil.

bending modes, respectively, with natural frequencies ranging from 9.54 Hz for the first bending to 118.11 Hz for the third bending mode. Fig. 18 displays the first four modes and corresponding frequencies due to their significance. Published results show, for the AGARD 445.6 wing, that the first mode is the dominant mode and is usually the flutter mode, at least in the subsonic and transonic regimes.

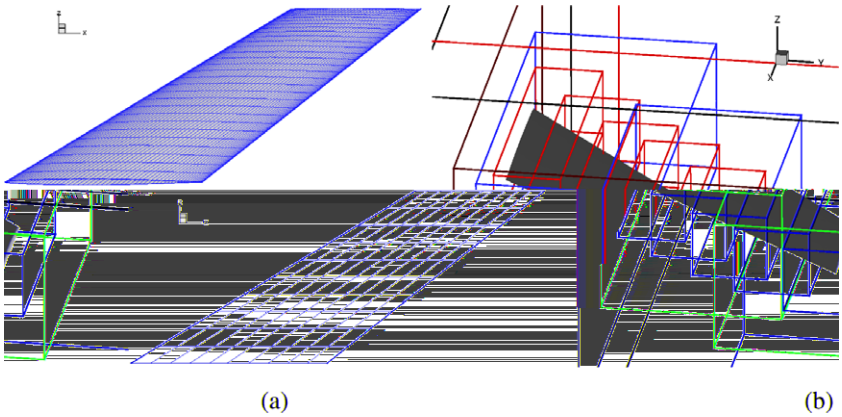


Fig. 17. Grid structure for the 445.6 wing: (a) surface grid and structural mode grid; (b) embedded grid blocks.

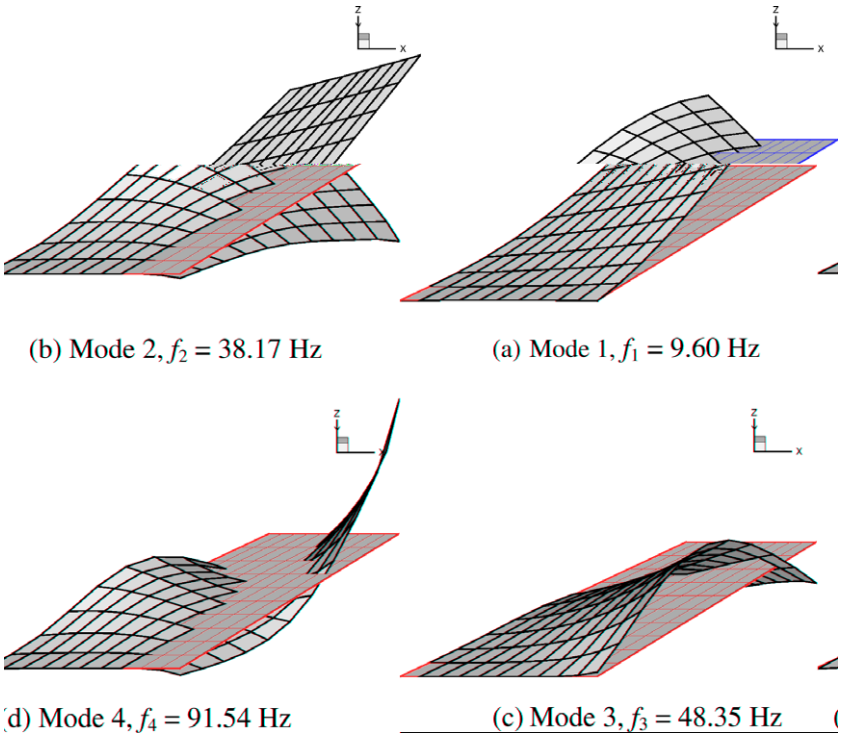


Fig. 18. Modal deflections for the AGARD 445.6 wing.

The present Cartesian-based numerical scheme resolves the unsteady flow by using stationary Cartesian grids in conjunction with the small perturbation method. For comparison, we also perform computations separately by using a body-fitted aeroelastic code, ParCAE, which is based on body-fitted curvilinear grids and the dynamic moving grid technique [25]. In the body-fitted-based approach, the CFD solver is based on a multiblock, multigrid finite-volume algorithm for the Euler equations. A multiblock deformation grid method is used to generate dynamically moving grids for the unsteady flow solver, and the BEM-based CFD–CSD interaction algorithm is applied to communicate data between fluid and structure systems.

Flutter computations for the AGARD 445.6 wing are performed over a range of Mach numbers in the transonic regime. We present here the computational results at two typical flight conditions: the first one is at Mach number $M_\infty = 0.96$ and speed index $v_f = 0.2$, where the wing is known to be stable; and the second one at $M_\infty = 1.141$ and $v_f = 0.75$, where the wing exhibits flutter. The time-histories of generalized displacements for the first flight condition are presented in Fig. 19(a) and (b), and those for the second flight condition are shown in Fig. 20(a) and (b). The computational results obtained with the present Cartesian-based approach and those with the body-fitted-moving-grid approach are compared here. It is noticed

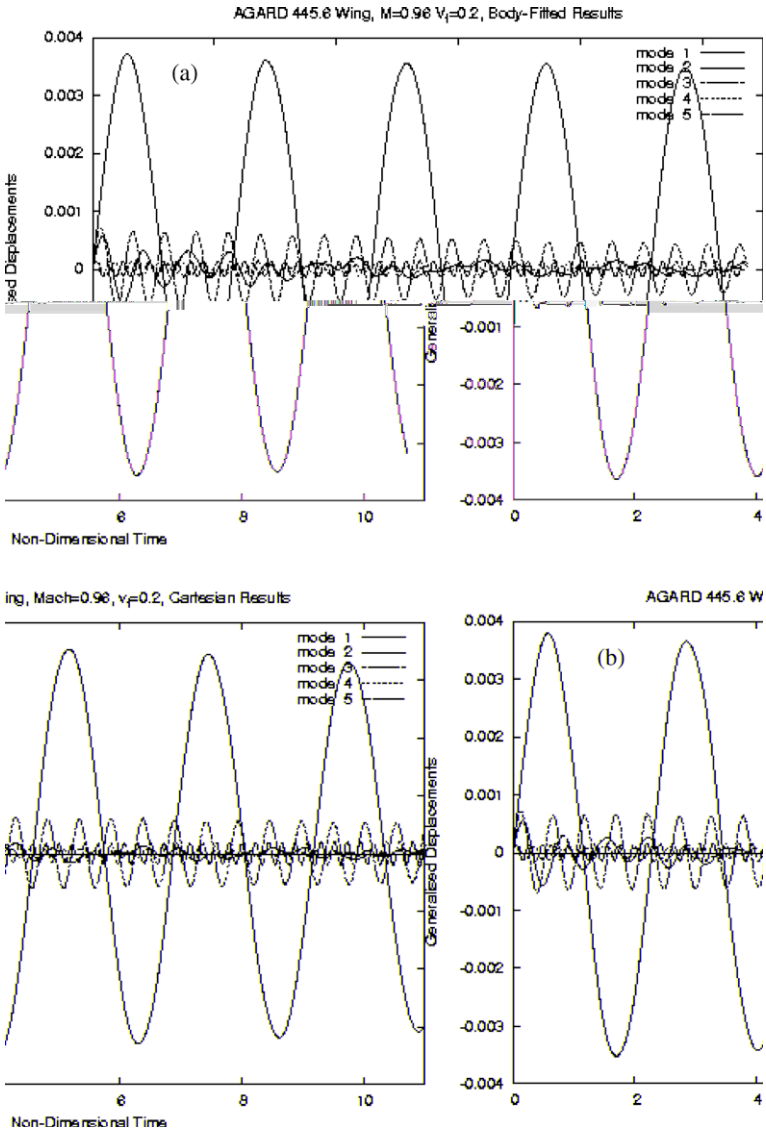


Fig. 19. Time history of the generalized coordinates with $M_\infty = 0.96$ and $v_f = 0.2$ for the AGARD 445.6 wing: (a) body-fitted grid results; (b) Cartesian grid results.

that the results of both approaches show almost identical trends of flutter behavior with respect to the amplitude and frequency of oscillation. The amplitude of oscillation of the first mode is considerably higher than the other modes, indicating that the first mode is the dominant flutter mode for both flight conditions. For the first flight condition, the system oscillates with a slightly decaying amplitude, which suggests that the wing is stable albeit close to the flutter boundary at that flight condition. For the second flight condition, Fig. 20(a) and (b) show unstable flutter behavior – the generalized displacement of the dominant mode oscillates at a very small amplitude initially but then grows exponentially.

By performing the above time-domain simulations at different speed index for a given flight Mach number, we can determine the critical flutter speed index, above which the wing exhibits growing oscillations. The flutter boundary and frequency of the weakened wing over the Mach number range from 0.5 to 1.141 considered experimentally in [23] are thus determined and shown in Fig. 21 along with results by using a moving body-fitted curvilinear grid [19] and the experimental results [24]. Both methods agree very well with the experimental data in the subsonic and transonic range. In the supersonic range, however, both methods produce much higher values than the experimental values consistent with other computations as discussed in [26]. The results by the present stationary Cartesian grid method almost overlap with those by the method using moving body-fitted grid over the complete Mach number range. The advantages of using a stationary Cartesian grid in terms of saving both human labor and computational resources make the present method much more desirable for engineering applications.

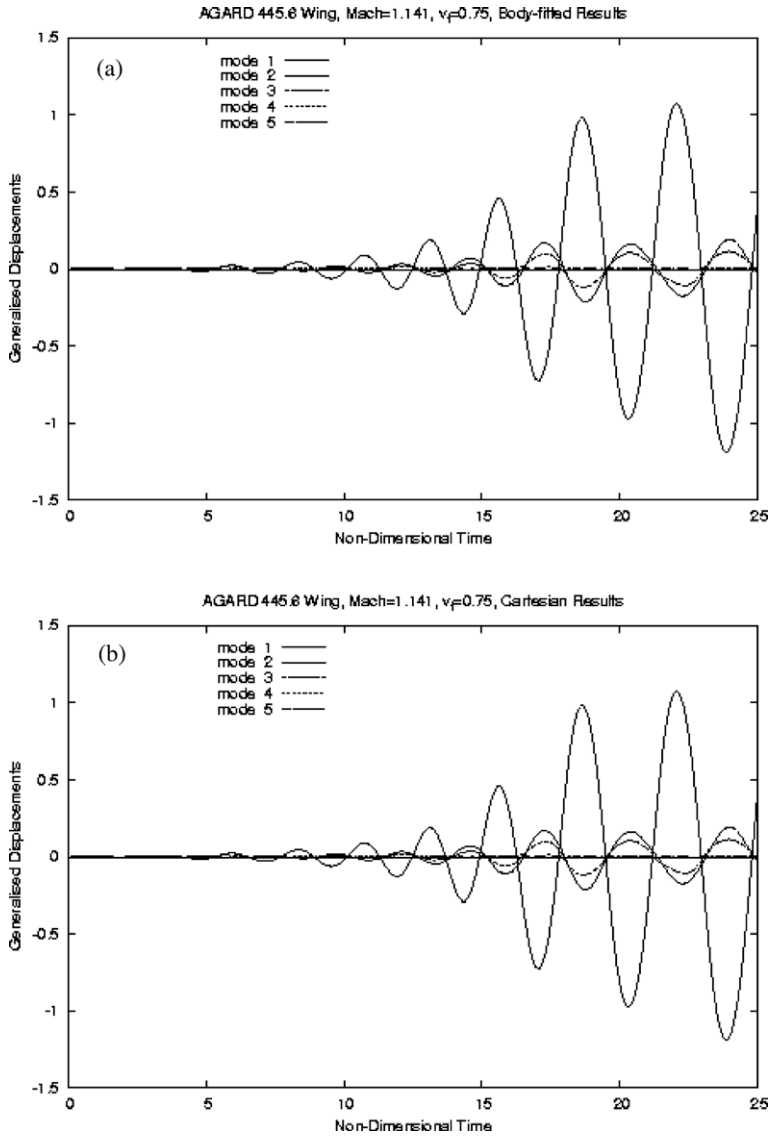


Fig. 20. Time history of the generalized coordinates with $M_\infty = 1.141$ and $v_f = 0.75$ for the AGARD 445.6 wing: (a) body-fitted grid results; (b) Cartesian grid results.

4. Conclusion

A three-dimensional computational method using stationary Cartesian mesh with embedded multigrid strategy is presented for steady and unsteady transonic flow computations of complex geometries. The proposed method eliminates the difficult task of generating body-fitted grid for complex geometries and the need for deforming and moving the grid for unsteady computations.

The method makes use of reflected points off a solid wall and implements a least-squares gridless interpolation to implement wall boundary conditions. The resulting method is computationally efficient and easy to implement. To provide needed accuracy, a multi-level embedded grid method is used to refine the grid successively where high resolution is required. The multi-level successively embedded grid also provides the grid levels for implementing a multigrid acceleration technique. Furthermore, to compute unsteady flows with moving boundaries, a small-perturbation method for the unsteady wall boundary condition is implemented to eliminate the use of moving grids that often cause loss of efficiency and lack of robustness of the computations. The use of the Cartesian grid, the innovative gridless method for treating wall boundary conditions, the small-perturbation unsteady wall boundary condition method, and the multi-level embedded mesh for both local refinement and multigrid acceleration, combine to make the present method efficient, accurate, and easy to use for practical three-dimensional problems with complex geometry.

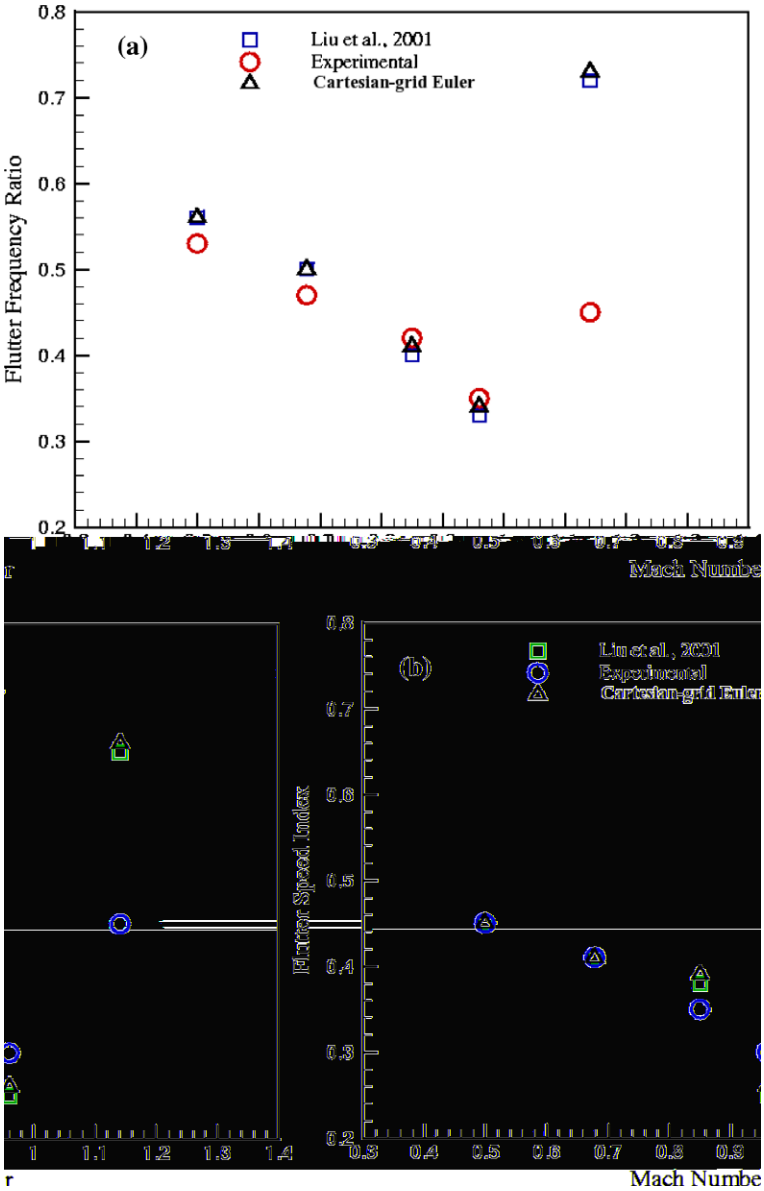


Fig. 21. Flutter frequency (left) and flutter speed (right) for the AGARD 445.6 wing: (a) flutter frequency ratio; (b) flutter speed index.

Test cases of airfoils, wings, and wing-body combinations and flutter computations of the AGARD 445.6 wing demonstrate that the method provides equal computational accuracy to those by another body-fitted grid method compared to experimental data for both steady and unsteady flows. The proposed small-perturbation boundary condition method using non-moving Cartesian grids is particularly advantageous for the large number of flutter simulations often needed for different aircraft configurations and flight conditions where minimum effort of human labor as well as computer time is immensely desirable.

For future work one can explore the incorporation of an automated grid embedment and refinement process based on a flow adaptive grid strategy. Extension to viscous computation is also another challenge, especially for high Reynolds number flows. One approach is to couple the Cartesian grid approach with an interactive boundary-layer method as demonstrated in [27].

Acknowledgments

The authors are grateful to the reviewers whose comments have helped us improve the manuscript considerably.

References

[1] D.J. Kirshman, F. Liu, A gridless boundary condition method for the solution of the Euler equations on embedded Cartesian meshes with multigrid, Journal of Computational Physics 201 (1) (2004) 119–147.
[2] D.K. Clarke, M.D. Salas, H.A. Hassan, Euler calculations for multi-element airfoils using Cartesian grids, AIAA Journal 24 (3) (1986) 353–358.
[3] M.J. Berger, R.J. LeVeque, An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries, AIAA Paper 89-1930, the 9th AIAA Computational Fluid Dynamics Conference, Buffalo, NY, June 13–15, 1989.
[4] W.J. Coirier, K.G. Powell, An accuracy assessment of Cartesian mesh approaches for the Euler equations, Journal of Computational Physics 117 (1) (1995) 121–131.
[5] E.P.C. Koh, H.M. Tsai, F. Liu, Euler solution using Cartesian grid with a gridless least squares boundary treatment, AIAA Journal 43 (2) (2005) 246–255.
[6] K. Lai, E.P.C. Koh, H.M. Tsai, Flutter computations of complex configurations using Cartesian grids, AIAA paper 2006-1399, the 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 9–12, 2006.
[7] L. Carolina, H.M. Tsai, F. Liu, An embedded Cartesian grid Euler solver with radial basis function for boundary condition implementation, AIAA paper 2008-532, the 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 7–10, 2008.
[8] H.M. Tsai, A.S.F. Wong, J. Cai, Y. Zhu, F. Liu, Unsteady flow calculations with a multi-block moving mesh algorithm, AIAA Journal 39 (6) (2001) 1021–1029.
[9] C. Gao, S. Yang, S. Luo, F. Liu, D.M. Schuster, Calculation of airfoil flutter by an Euler method with approximate boundary conditions, AIAA Journal 43 (2) (2005) 295–305.
[10] S. Yang, F. Liu, S. Luo, H.M. Tsai, D.M. Schuster, Time-domain aeroelastic simulation on stationary body-conforming grids with small perturbation boundary conditions, AIAA Paper 2004-0885, the 42nd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 5–8, 2004.
[11] S. Yang, F. Liu, S. Luo, H.M. Tsai, Three-dimensional aeroelastic computation based on stationary body-conforming grids with small perturbation boundary conditions, AIAA paper 2004-2235, the 34th AIAA Fluid Dynamics Conference and Exhibit, Portland, Oregon, June 28–July 1, 2004.
[12] D.J. Kirshman, F. Liu, Flutter prediction by an Euler method on non-moving Cartesian grid with gridless boundary conditions, Computers and Fluids 35 (6) (2006) 571–586.
[13] A. Jameson, W. Schmidt, E. Turkel, Numerical solutions of the Euler equations by finite volume methods using Runge–Kutta time-stepping schemes, AIAA Paper 81-1259, the 14th Fluid and Plasma Dynamics Conference, Palo Alto, CA, June 23–25, 1981.
[14] A. Jameson, Multigrid algorithms for compressible flow calculations, Rept. 1743, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, October 1985.
[15] A. Jameson, Time-dependent calculations using multigrid with applications to unsteady flows past airfoils and wings, AIAA Paper 1991-1596, June 1991.
[16] F. Liu, S. Ji, Unsteady flow calculations with a multigrid Navier–Stokes method, AIAA Journal 34 (10) (1996) 2047–2053.
[17] A. Jameson, Solution of the Euler equations for two dimensional transonic flow by a multigrid method, Applied Mathematics and Computation 13 (1983) 327–355.
[18] A. Dadone, B. Grossman, Surface boundary conditions for the numerical solution of the Euler equations, AIAA Journal 32 (2) (1994) 285–293.
[19] F. Liu, J. Cai, Y. Zhu, A.S.F. Wong, H.M. Tsai, Calculation of wing flutter by a coupled fluid–structure method, Journal of Aircraft 38 (2) (2001) 334–342.
[20] V. Schmitt, F. Charpin, Pressure distributions on the ONERA-M6-Wing at transonic Mach numbers, Experimental Data Base for Computer Program Assessment, AGARD Advisory Report AR-138, 1979.
[21] D.A. Treasogold, A.F. Jones, K.H. Wilson, Pressure distributions measured in the RAE 8ft $\times$ 6ft transonic wing tunnel on RAE wing a in combination with an axisymmetric body at Mach numbers of 0.4, 0.8, and 0.9, Chap B-4, AGARD Advisory Report AR-138, May 1989.
[22] R.H. Landon, NACA 0012. Oscillatory and transient pitching, Compendium of Unsteady Aerodynamics Measurements, AGARD R-702, August 1982.
[23] E.C. Yates Jr., N.S. Land, J.T. Foughner Jr., Measured and calculated subsonic and transonic flutter characteristics of a 45 degree sweptback wing planform in air and Freon-12 in the Langley Transonic Dynamics Tunnel, NASA TN D-1616, March 1963.
[24] E.C. Yates Jr., AGARD standard aeroelastic configurations for dynamic response I – Wing 445.6, NASA TM 100492, August 1987.
[25] M. Sadeghi, S. Yang, F. Liu, Parallel computation of wing flutter with a coupled Navier–Stokes/CSD method, AIAA Paper 2003-1347, January 2003.
[26] E.M. Lee-Rausch, J.T. Batina, Wing flutter boundary prediction using unsteady Euler aerodynamic method, Journal of Aircraft 32 (2) (1995) 416–422.
[27] Z. Zhang, F. Liu, D.M. Schuster, Calculations of unsteady flow and flutter by an Euler and integral boundary-layer method on Cartesian grids, AIAA paper 2004-5203, the 22nd Applied Aerodynamics Conference and Exhibit, Providence, Rhode Island, August 16–19, 2004.