AIAA JOURNAL
Vol. 39, No. 6, June 2001

Unsteady Flow Calculations with a Parallel Multiblock Moving Mesh Algorithm

H. M. Tsai* and A. S. F. Wong†
Defence Science Organization National Laboratories, Singapore 118230, Republic of Singapore
and
J. Cai,‡ Y. Zhu,§ and F. Liu¶
University of California, Irvine, Irvine, California 92697-3975

A novel parallel dynamic moving mesh algorithm designed for multiblock parallel unsteady flow calculations using body-fitted grids is presented. The moving grid algorithm within each block uses a method of arc-length-based transfinite interpolation, which is performed independently on local processors where the blocks reside. A spring network approach is used to determine the motion of the corner points of the blocks, which may be connected in an unstructured fashion in a general multiblock method. A smoothing operator is applied to the points of the block face boundaries and edges to maintain grid smoothness and grid angles. A multiblock parallel Euler/Navier–Stokes solver using multigrid and dual-time stepping is developed along with the moving mesh method. Computational results are presented for the unsteady flow calculations of airfoils and wings with deforming shapes as found in flutter simulations.

Introduction

UNSTEADY flow calculations for problems with moving boundaries such as those found in the flutter simulation of wings and turbomachinery blades require a fast and reliable method for deforming the computational grid as the flow boundary changes shape.[1–7] Efficient deforming grid algorithms are also needed for aerodynamic shape optimization studies.[8,9] Regenerating a grid at each time step in an unsteady calculation or in each design cycle in a design optimization program is a natural choice. However, generation of a grid for a complex geometry is by itself a nontrivial and time-consuming task for the user. Because the grid must be updated frequently in an unsteady or design optimization computation, an automatic and fast algorithm to deform an existing grid is needed. It is also desirable that the dynamically moving grid retains the essential qualities of an initial grid generated by, possibly, a sophisticated grid-generation package.

A spring-analogy method initially proposed by Batina[10] is used by Lee-Rausch and Batina[2] and Robinson et al.[11] to generate dynamic grids for both structured and unstructured solvers. The same method is used by Prananta and Hounjet.[6] The spring-analogy method can adapt to large surface deformations and is applicable to both structured and unstructured grids. However, it is an iterative method resembling an elliptic grid generator. Therefore, it is computationally time consuming for large numbers of grid points.

In the flutter simulation codes by Bhardwaj et al.,[3] Byun and Guruswamy[4] and Ji and Liu,[5] simple algebraic methods are used to deform an existing grid by redistributing points along grid lines that are in the direction normal to the surface. This type of method is very fast but works only for small deformations. Large deformations may produce grids of poor quality or crossover of grid lines. In addition, for complex geometries, it may not always be straightforward to identify a radial direction normal to the structural surfaces.

Jones and Samareh-Abolhassani[9] and Reuther et al.[8] developed structured grid-generation systems for use in multidisciplinary design optimization based on variations of the transfinite interpolation (TFI) method. These are algebraic methods that automatically handle multiple moving surfaces and also take into account the original interior grid distribution in moving the grid points. Chen et al.[7] also use such a method for aeroelastic analysis. Multiblocks can be used in this method, but grid motion of the block surfaces must be determined first. A challenge also exists in a multiblock approach where blocks are connected in an unstructured fashion. Furthermore, parallel computation is inevitably necessary for large computational problems, wherein the blocks are usually distributed over different processors for parallel processing in a domain decomposition approach. Therefore, a grid-deformation method should allow deformation to be accomplished in parallel on each processor without having to assemble all of the blocks on one processor and with little communication between processors. These issues have not been discussed in the current literature.

The purpose of this paper is to develop a new grid-deformation method that combines the advantages of existing methods currently used in aeroelasticity and design optimization codes while avoiding their shortcomings. Specifically, a dynamic moving mesh algorithm is developed that is 1) flexible in handling arbitrary moving surfaces, 2) retains basic features of an initial grid, 3) capable of handling multiple blocks of structured grids connected in either structured or unstructured fashion, 4) fast by using an algebraic method within each block, and 5) naturally parallelizable in a multiblock parallel flow solver without having to assemble the blocks and with minimum interprocessor communications.

To achieve those goals, a combination of algebraic and iterative methods is used. The motion of all corner points is first determined by using a spring network analogy as proposed by Batina.[10] The locations of all of the block corner points of the complete domain are stored on one processor. These corner points are connected by springs. When points move that are on a moving flow boundary such as the body of an oscillating wing, other corner points are moved accordingly based on the balance of the spring network. In general, the number of blocks and, thus, the number of block corner points are limited and far fewer than the volume grid points in the flowfield. This spring analogy procedure can be computed very quickly and can be performed on one single processor where the locations of all of the corner points are stored.

Received 18 February 2000; revision received 20 December 2000; accepted for publication 21 December 2000. Copyright © 2001 by the authors. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

*Senior Member of Technical Staff, Aeronautical Systems Program; currently Adjunct Associate Professor, National University of Singapore, Singapore 119260, Republic of Singapore. Member AIAA.

†Senior Member of Technical Staff, Aeronautical Systems Program, 20 Science Park Drive.

‡Visiting Research Specialist, Department of Mechanical and Aerospace Engineering; Associate Professor, Department of Aerospace Engineering, Northwestern Polytechnical University, Xi'an 710072, People's Republic of China.

§Graduate Student, Department of Mechanical and Aerospace Engineering. Member AIAA.

¶Associate Professor, Department of Mechanical and Aerospace Engineering. Senior Member AIAA.

Once the new locations of the corner points are determined on this processor, they are sent to other processors, where each individual block belongs. On receiving the block corner locations of a block belonging to a particular processor, this processor will then perform an algebraic procedure using an arclength-based TFI for the grid deformation (not the grid itself) to determine the edge, face, and ultimately volume grid points for the entire block in parallel without the need of any communication with other processors. A method to maintain grid continuity at the block to block interfaces is introduced based on an extension of the blending function used by Tavella[12] to orthogonalize algebraically the grid near solid surfaces.

The basic formulation of this new deforming grid method along with the unsteady flow solver used for flutter calculations is presented in the following sections. Computational results are presented to demonstrate that the method is fast and robust for unsteady flow over oscillating and deforming airfoils and wings. Large-amplitude vibration of the wing can be studied without the possibility of crossover cells.

Parallel Multiblock Navier–Stokes Solver

A multigrid and multiblock method for the solution of the three-dimensional unsteady compressible Euler and Navier–Stokes equations of a body in arbitrary motion and deformation using parallel computers is developed. The flow solver solves the Euler/Navier–Stokes equations with a finite volume method. The program is based on an existing steady single-processor Navier–Stokes code, NSAERO, originally developed by Tsai et al.[13] Both Jameson–Schmidt–Turkel[14] and CUSP[15,16] types of schemes are implemented for spatial discretization. The unsteady extension uses an implicit second-order time-accurate scheme for time discretization. The implicit equations are then solved by using a dual-time-stepping algorithm with multigrid.[17–19]

To take advantage of parallel processing, the flowfield is partitioned into multiple blocks, which are distributed over a number of processors available on a parallel computer or networked personal computers. The dual-time implicit–explicit solver is performed on each processor for the individual blocks assigned to the processor. In the present work, the partitioning strategy places the block loop within the multigrid cycle so that accuracy and convergence are not adversely affected by the artificially generated block boundaries. Two layers of halo cells are used beyond the boundaries of each block to facilitate the implementation of boundary conditions and the communication between processors. More details of the parallel unsteady solver is provided in Refs. 20 and 21.

Grid Deformation Method

Simple algebraic methods, although relatively inexpensive to run, are limited to small-amplitude motions. Large motions could cause the grid lines to coalesce or crossover. Iterative methods (such as those treating the entire grid domain as connected by springs) can handle large motions but are hampered by expensive CPU cycles. The deforming grid code developed here, algebraic and iterative mesh 3D (AIM3D) is based on a combination of algebraic and iterative methods. This code combines the robustness of an iterative method with the speed of an algebraic method. It also allows for automatic remeshing of multiblock grids without recourse to interblock communication. In a single-block mode, AIM3D uses an arclength-based TFI to generate or modify block faces and interior volume grids. In a multiblock mode, AIM3D uses a spring analogy to determine the eight corner points of each block. This method is iterative in nature, using Jacobi iterations for the spring static equilibrium equations. Various other features were implemented in AIM3D to preserve the original undeformed grid quality while attempting, as far as possible, to maintain robustness and increase versatility.

Single-Block Approach

An arclength-based TFI method is used when treating a single block. The process follows that proposed by Jones and Samareh-Abolhassani[9] and can be broken down into several steps:

1) Parameterize all grid points on a surface.
2) Compute all edge and corner point deformations.
3) Perform one-, two-, and three-dimensional TFI of the deformations.
4) Add the deformations to the original grid to get the final grid.

In the parameterization process, all grid points are parameterized according to the global i, j, and k indices of the arclengths. This is accomplished as follows. First, all of the arclengths on a surface are computed by adding up all of the displacements along a grid line. This is done by keeping the other two indices constant and varying the third. For example, the arclength parameter in the i direction for any fixed j and k surface is

$$s_{1,j,k} = 0 \tag{1}$$

$$s_{i,j,k} = s_{i-1} + \sqrt{(x_{i,j,k} - x_{i-1,j,k})^2 + (y_{i,j,k} - y_{i-1,j,k})^2 + (z_{i,j,k} - z_{i-1,j,k})^2} \tag{2}$$

$$\text{for } i = 2, \ldots, i\max \tag{3}$$

where $i\max$ is the maximum grid index in the i direction. The normalized arclength parameter for this grid line is then defined as

$$F_{i,j,k} = s_{i,j,k}/s_{i\max,j,k} \tag{4}$$

Similarly, the grid parameters for the grid lines in the j and k directions can be defined, which we shall denote by $G_{i,j,k}$ and $H_{i,j,k}$, respectively.

The next stage is to compute the edge and corner point deformations for the newly deformed body. Based on these known deformations, one-, two-, or three-dimensional TFI is used to create interpolated values of the deformations, which are then added to the original grid to create a new volume grid that maintains the grid quality of the original grid and maximizes the reusability of the original grid. The one-dimensional TFI in the k direction is simply

$$\Delta E_{1,1,k} = (1 - H_{1,1,k})\Delta P_{1,1,1} + H_{1,1,k}\Delta P_{1,1,k\max} \tag{5}$$

where ΔE refers to the deformation of an edge, which in this example varies in the k index, whereas $\Delta P_{1,1,1}$ and $\Delta P_{1,1,k\max}$ are the deformations of the two corner points of the edge. The two-dimensional TFI is defined as (for a surface in the $i = 1$ plane)

$$\begin{aligned}\Delta S_{1,j,k} &= A_{1,j,k}\Delta E_{1,j,1} + B_{1,j,k}\Delta E_{1,j,k\max} + C_{1,j,k}\Delta E_{1,1,k} \\ &+ D_{1,j,k}\Delta E_{1,j\max,k} - A_{1,j,k}C_{1,j,k}\Delta P_{1,1,1} \\ &- B_{1,j,k}C_{1,j,k}\Delta P_{1,1,k\max} - A_{1,j,k}D_{1,j,k}\Delta P_{1,j\max,1} \\ &- B_{1,j,k}D_{1,j,k}\Delta P_{1,j\max,k\max}\end{aligned} \tag{6}$$

where ΔS, ΔE, and ΔP refer to the surface, edge, and corner point deformations, respectively. The subscripts are the indices of the grid and span between (1, 1, 1) and ($i\max$, $j\max$, and $k\max$). A robust set of blending functions proposed by Soni[22] is used:

$$\begin{aligned} A_{1,j,k} &= 1 - \eta_{1,j,k}, & B_{1,j,k} &= \eta_{1,j,k} \\ C_{1,j,k} &= 1 - \xi_{1,j,k}, & D_{1,j,k} &= \xi_{1,j,k} \end{aligned} \tag{7}$$

where

$$\begin{aligned} P_{1,j,k} &= 1 - (G_{1,j,k\max} - G_{1,j,1})(H_{1,j\max,k} - H_{1,1,k}) \\ \xi_{1,j,k} &= [G_{1,j,1} + H_{1,1,k}(G_{1,j,k\max} - G_{1,j,1})]/P_{1,j,k} \\ \eta_{1,j,k} &= [H_{1,1,k} + G_{1,j,1}(H_{1,j\max,k} - H_{1,1,k})]/P_{1,j,k} \end{aligned} \tag{8}$$

A standard three-dimensional TFI formula is used to determine the deformation of all of the points in the volume grid:

$$\Delta V_{i,j,k} = V1 + V2 + V3 - V12 - V13 - V23 + V123 \tag{9}$$

where

$$V1 = (1 - F_{i,j,k})\Delta S_{1,j,k} + F_{i,j,k}\Delta S_{i\,\max,j,k}$$

$$V2 = (1 - G_{i,j,k})\Delta S_{i,1,k} + G_{i,j,k}\Delta S_{i,j\,\max,k}$$

$$V3 = (1 - H_{i,j,k})\Delta S_{i,j,1} + H_{i,j,k}\Delta S_{i,j,k\,\max}$$

$$\begin{aligned} V12 &= (1 - F_{i,j,k})(1 - G_{i,j,k})\Delta E_{1,1,k} + (1 - F_{i,j,k})G_{i,j,k}\Delta E_{1,j\,\max,k} \\ &\quad + F_{i,j,k}(1 - G_{i,j,k})\Delta E_{i\,\max,1,k} + F_{i,j,k}G_{i,j,k}\Delta E_{i\,\max,j\,\max,k} \\ V13 &= (1 - F_{i,j,k})(1 - H_{i,j,k})\Delta E_{1,j,1} + (1 - F_{i,j,k})H_{i,j,k}\Delta E_{1,j,k\,\max} \\ &\quad + F_{i,j,k}(1 - H_{i,j,k})\Delta E_{i\,\max,j,1} + F_{i,j,k}H_{i,j,k}\Delta E_{i\,\max,j,k\,\max} \\ V23 &= (1 - G_{i,j,k})(1 - H_{i,j,k})\Delta E_{i,1,1} + (1 - G_{i,j,k})H_{i,j,k}\Delta E_{i,1,k\,\max} \\ &\quad + G_{i,j,k}(1 - H_{i,j,k})\Delta E_{i,j\,\max,1} + G_{i,j,k}H_{i,j,k}\Delta E_{i,j\,\max,k\,\max} \end{aligned} \tag{10}$$

$$\begin{aligned} V123 &= (1 - F_{i,j,k})(1 - G_{i,j,k})(1 - H_{i,j,k})\Delta P_{1,1,1} \\ &\quad + (1 - F_{i,j,k})(1 - G_{i,j,k})H_{i,j,k}\Delta P_{1,1,k\,\max} \\ &\quad + (1 - F_{i,j,k})G_{i,j,k}(1 - H_{i,j,k})\Delta P_{1,j\,\max,1} \\ &\quad + (1 - F_{i,j,k})G_{i,j,k}H_{i,j,k}\Delta P_{1,j\,\max,k\,\max} \\ &\quad + F_{i,j,k}(1 - G_{i,j,k})(1 - H_{i,j,k})\Delta P_{i\,\max,1,1} \\ &\quad + F_{i,j,k}(1 - G_{i,j,k})H_{i,j,k}\Delta P_{i\,\max,1,k\,\max} \\ &\quad + F_{i,j,k}G_{i,j,k}(1 - H_{i,j,k})\Delta P_{i\,\max,j\,\max,1} \\ &\quad + F_{i,j,k}G_{i,j,k}H_{i,j,k}\Delta P_{i\,\max,j\,\max,k\,\max} \end{aligned} \tag{11}$$

Multiblock Approach

In a multiblock context, the blocks may be arranged in an unstructured fashion relative to each other. The motion of the eight corner points of each block cannot be readily determined by using the algebraic methods. Therefore, the concept of springs as proposed in Ref. 10 is adopted but used here only for the block corner points in a multiblock grid solver. The corner points of all of the blocks are assumed to be connected by springs with the spring stiffness being inversely proportional to the length of the connecting edges:

$$\begin{aligned} k_{i+\frac{1}{2},j,k} &= 1\Big/\Big[(x_{i+1,j,k} - x_{i,j,k})^2 + (y_{i+1,j,k} - y_{i,j,k})^2 \\ &\quad + (z_{i+1,j,k} - z_{i,j,k})^2\Big]^{p/2} \end{aligned} \tag{12}$$

For simplicity, the length based on the distance between the corner points is used instead of the arclength of the edge connecting the corner points. Spring stiffness is computed for all 12 edges of a block plus additional 4 cross-diagonal edges running through the interior of the block. These cross-diagonal edges are for additional control over the shearing motion of the grid cells. The power p is used as by Robinson et al.[11] to control the stiffness of the grid cells, in this case, blocks. The higher the value of p, the stiffer the blocks are. Typically, a value of 1 is used.

With the prescribed motion of the deformed surfaces, the displacements of the corner points of each block are determined by solving the equations of static equilibrium using a predictor–corrector iterative process.

Predictor step:

$$\begin{aligned} (\overline{\delta x})_{i,j,k} &= 2(\delta x)^n_{i,j,k} - (\delta x)^{n-1}_{i,j,k} \\ (\overline{\delta y})_{i,j,k} &= 2(\delta y)^n_{i,j,k} - (\delta y)^{n-1}_{i,j,k} \\ (\overline{\delta x})_{i,j,k} &= 2(\delta z)^n_{i,j,k} - (\delta z)^{n-1}_{i,j,k} \end{aligned} \tag{13}$$

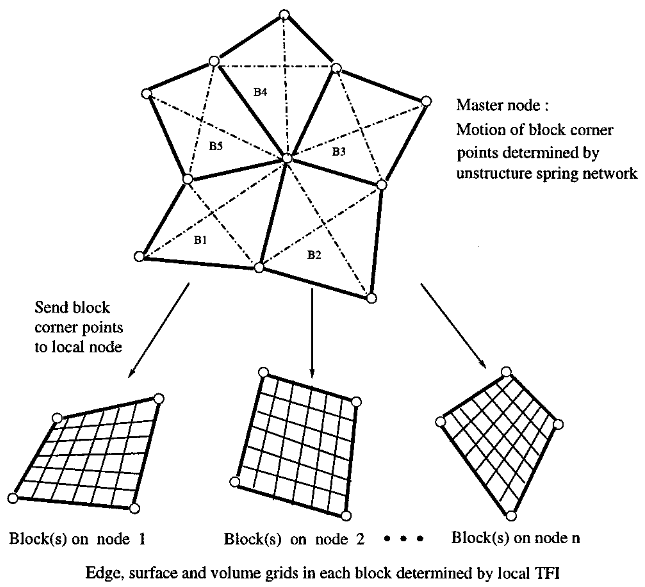
B4
B5
B3
B1
B2
Master node :
Motion of block corner points determined by unstructure spring network
Send block corner points to local node
Block(s) on node 1
Block(s) on node 2 • • •
Block(s) on node n
Edge, surface and volume grids in each block determined by local TFI

Fig. 1 Parallel multiblock deformation strategy.

Corrector step:

$$\begin{aligned} (\delta x)^{n+1}_{i,j,k} &= \frac{k_{i+\frac{1}{2},j,k}(\overline{\delta x})_{i,j,k} + \cdots + k_{i,j,k-\frac{1}{2}}(\overline{\delta x})_{i,j,k-1}}{k_{i+\frac{1}{2},j,k} + \cdots + k_{i,j,k-\frac{1}{2}}} \\ (\delta y)^{n+1}_{i,j,k} &= \frac{k_{i+\frac{1}{2},j,k}(\overline{\delta y})_{i,j,k} + \cdots + k_{i,j,k-\frac{1}{2}}(\overline{\delta y})_{i,j,k-1}}{k_{i+\frac{1}{2},j,k} + \cdots + k_{i,j,k-\frac{1}{2}}} \\ (\delta z)^{n+1}_{i,j,k} &= \frac{k_{i+\frac{1}{2},j,k}(\overline{\delta z})_{i,j,k} + \cdots + k_{i,j,k-\frac{1}{2}}(\overline{\delta z})_{i,j,k-1}}{k_{i+\frac{1}{2},j,k} + \cdots + k_{i,j,k-\frac{1}{2}}} \end{aligned} \tag{14}$$

In an implementation using parallel computers, the described procedure is carried out on one processor. Once the coordinates of the eight corner points of all of the blocks are determined, they are sent to each processor, where the blocks belong. The grid motion for each block can then be generated using the algebraic method described in the preceding section on each individual processor. Figure 1 shows this multiblock multiprocessor parallel strategy. The advantage of this methodology is that TFI can be performed independently on each processor for the blocks that belong to the processor without having to store the complete set of blocks on each processor and with minimum communication needed between blocks and processors. Because the TFI is applied to only the displacement of grid points and the newly deformed grid is obtained by adding the interpolated displacements to the original grid points, the new grid retains much of the same features of the original grid, such as smoothness and orthogonality. In addition, the nature of the method guarantees automatic matching of edges and surfaces between blocks if the block-to-block interfaces match in the initial grid. The initial grid can be generated interactively by using a sophisticated grid-generation package.

Mixed Block Boundary Conditions

There are cases in which, because of grid topology or physical nature of the problem, only a certain portion(s) of an edge or surface is deformed. This can happen when part of an edge/surface is a wing and the rest of it is off-body grid points. Motion of the wing will result in discontinuities between the deformed and undeformed portions of the same edge or surface. To accommodate such mixed boundary conditions, a smoothing operator is applied to alleviate the discontinuities. For edge smoothing, the operator is simply a second-order differencing with weightings inversely proportional to the distances. For example, the following smoothing is used for an edge in the i direction:

$$P_i = \frac{P_{i+1} - P_i}{P_{i+1} - P_{i-1}}P_{i+1} + \frac{P_i - P_{i-1}}{P_{i+1} - P_{i-1}}P_{i-1} \tag{15}$$

where the subscript indices j and k are not shown for simplicity.

Following Kodama,[23] a Laplace-like operator is used for surface smoothing. This is a second-order differencing formula with weightings that are inversely proportional to the distances squared. For a $k = \text{const}$ surface, it is written

$$P_{i,j} = w_1(P_{i-1,j} + P_{i+1,j})/2 + w_2(P_{i,j-1} + P_{i,j+1})/2 \tag{16}$$

where the subscript k is neglected and

$$w_1 = \frac{(P_{i,j+1} - P_{i,j})(P_{i,j} - P_{i,j-1})}{(P_{i,j+1} - P_{i,j})(P_{i,j} - P_{i,j-1}) + (P_{i+1,j} - P_{i,j})(P_{i,j} - P_{i-1,j})} \tag{17}$$

$$w_2 = \frac{(P_{i+1,j} - P_{i,j})(P_{i,j} - P_{i-1,j})}{(P_{i,j+1} - P_{i,j})(P_{i,j} - P_{i,j-1}) + (P_{i+1,j} - P_{i,j})(P_{i,j} - P_{i-1,j})} \tag{18}$$

Grid Blending near Solid and Block Faces

At large deformations, crossover of cells may occur near the solid boundary or at sharp corners as in the trailing edge of a wing. To improve the grid quality near wall and prevent crossover of cells, a blending function was implemented such that it keeps the first grid point at the same angle with respect to the moving edge/surface as that in the original grid. The blending function is a weighting function of the arctangent that makes the deformed grid lines extend the same angle from the surface as the original line and is also approximately equal to the original line far away from the deformed surface.

The blending function is given as

$$F_1 = (2/\pi)\tan^{-1}[(i - I1)/TI2]^{pp} \tag{19}$$

$$F_2 = (2/\pi)\tan^{-1}[TI2/(i - I1)]^{pp} \tag{20}$$

where $TI2 = TI \times (I2 - I1)$, TI is the transition index and ranges from 0 to 1, $I2$ and $I1$ define the maximum and minimum grid index of the region where the blending is to occur, and pp is used to control the blending and is typically set to four. The blended values of a grid point are given by the sum of the product of F_1 with the nonblended grid point $P_{i,j,k}$ and the product of F_2 with the projection of the nonblended grid points in the angle subtended by the original grid $P^{\text{project}}_{i,j,k}$:

$$P^{\text{blend}}_{i,j,k} = F_2 P^{\text{project}}_{i,j,k} + F_1 P_{i,j,k} \tag{21}$$

Whereas this blending procedure is initially used for the grids near solid surfaces where the maintenance of orthogonality is important, the same method is also used for grids near block-to-block interfaces. This way, the grid continuity at the block to block boundaries could be maintained. The advantage of this procedure is that, if the original grid is continuous across the block-to-block boundary, continuity will be maintained without any recourse to using grid points of the halo cells, which one would need if we adopt some form of Laplacian smoothing. Thus, in the current procedure, no interprocessor communication is necessary in the entire grid remeshing stage. Typically the transition index TI is set to 2 or 3 points divided by the total number of grid points in the direction of the grid index emanating from the block face.

Computational Results

Testing of the Deformation Code

The steps required for the generation of a deformed grid is described next. First, a high-fidelity grid with the required surfaces and volume grid points is generated. This initial grid will provide the basis from which all subsequent deformed grids are created by AIM3D. Information concerning the topology of the grid in question is also needed. Specifically, the information regarding the surfaces and edges of each block is required by the program. This is simply whether the surface or edge in question is moving or stationary. If parts or portions of an edge/surface are moving and the rest are held fixed (mixed boundary conditions), the information will have to include the indexes of the pertinent edge/surface that are moving.

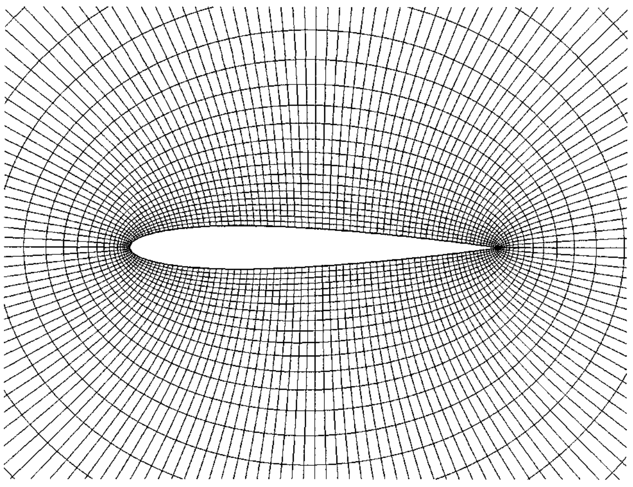
Fig. 2 NACA0012 mesh: one-block, close-up view.

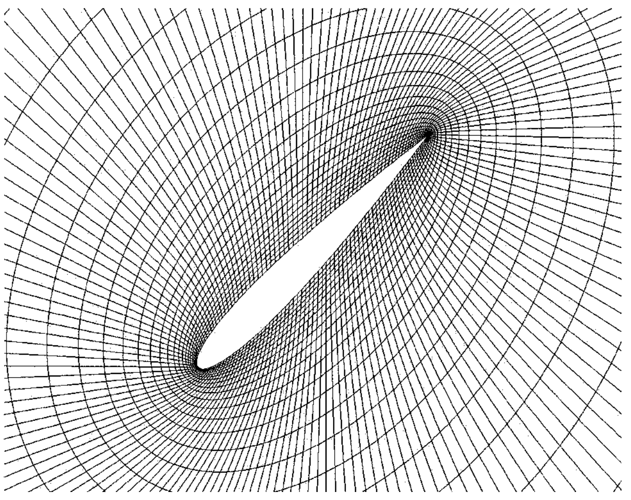
Fig. 3 NACA0012 mesh with 45-deg pitch down: one-block, close-up view.

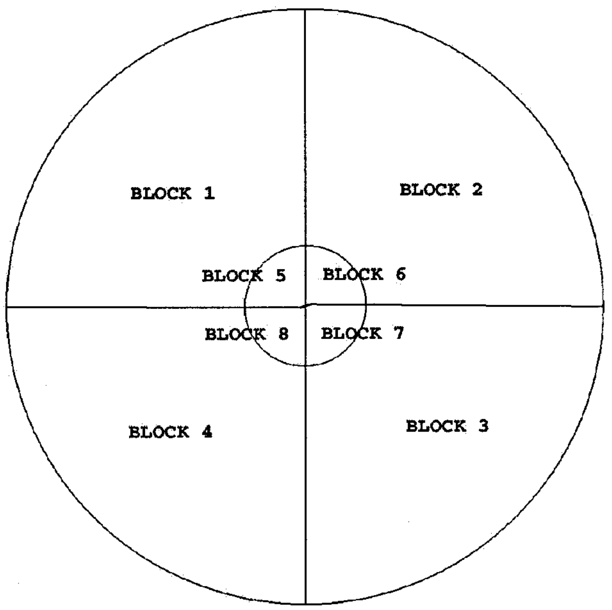

Fig. 4 Block boundaries: eight-block NACA0012 mesh.

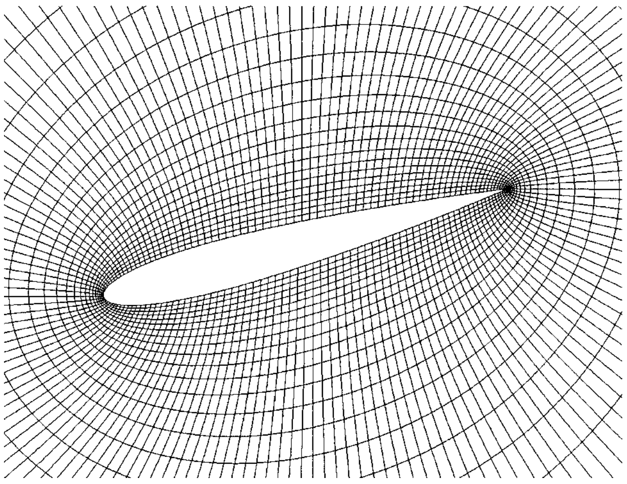

Fig. 5 Closeup view of eight-block NACA0012 with 20-deg pitch down.

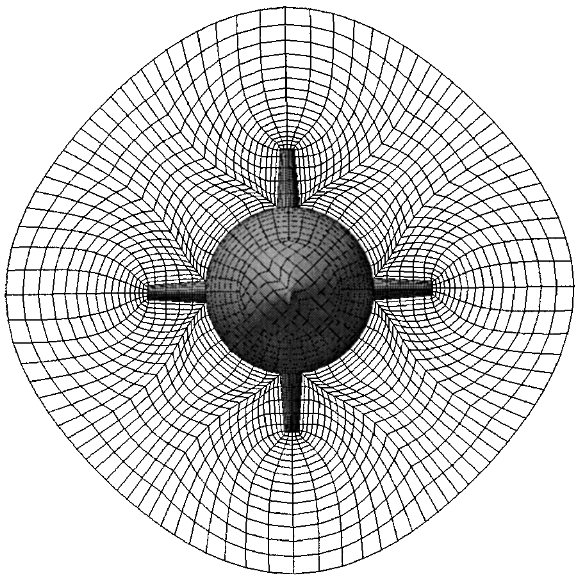

Fig. 8 Grid of a generic store with body and fin before deformation: 22 blocks.

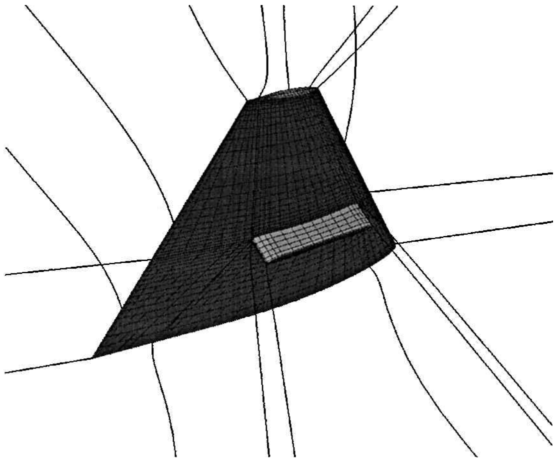

Fig. 6 Surface grid and block boundaries of a wing–pylon mesh.

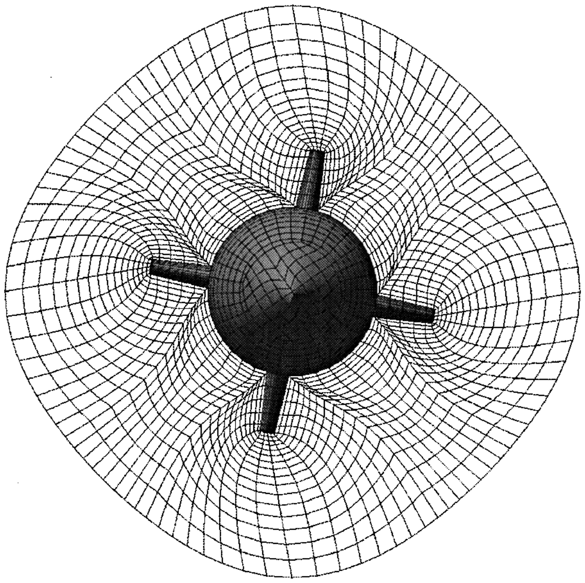

Fig. 9 Grid of a generic store with body and fin after 10-deg clockwise axial rotation: 22 blocks.

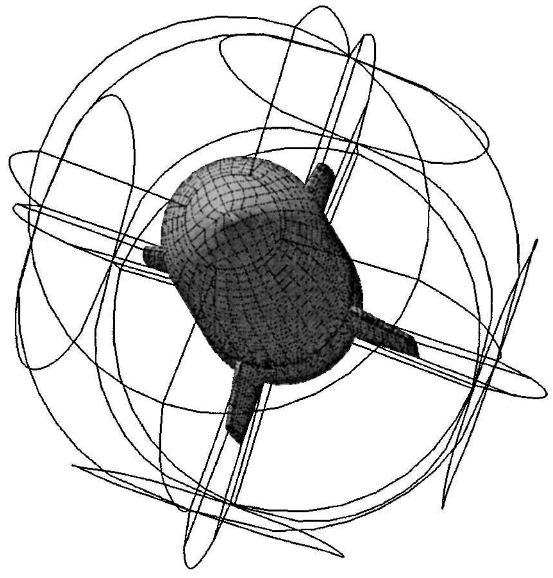

Fig. 7 Surface grid and block boundaries of a generic store with body and fin.

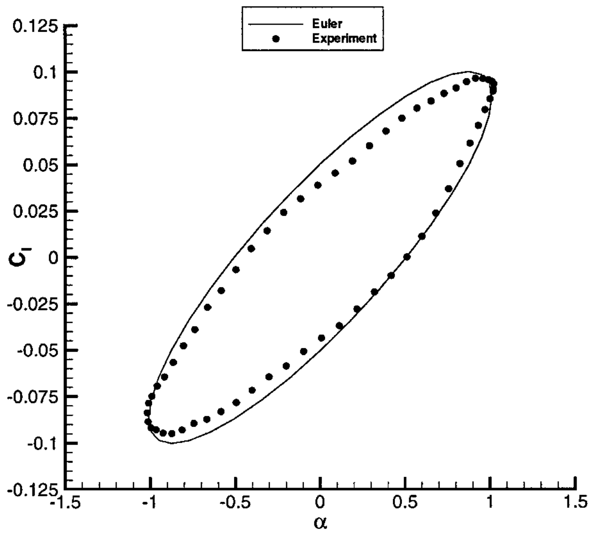

Fig. 10 Comparison of computed lift coefficient C_l vs α with experimental data (x–y plane).

Next, a grid needs to be generated with the prescribed surface deformations. AIM3D will generate a new grid that contains the same surface deformations and a deformed volume grid that while maintaining the grid quality of the undeformed volume grid also conforms to the deformed surfaces.

As far as possible, AIM3D uses existing information required for the computational fluid dynamics (CFD) solver. This helps to keep the number of AIM3D specific inputs to a bare minimum. For example, block-to-block connectivity information is required by the CFD solver and needs only be generated once by the CFD preprocessor. This connectivity information is also required in AIM3D, and it reads in the same connectivity file generated by the CFD preprocessor.

The following test cases demonstrate the usefulness of the proposed deformation method. First, a study was done on the performance of the code using the two-dimensional NACA0012 airfoil

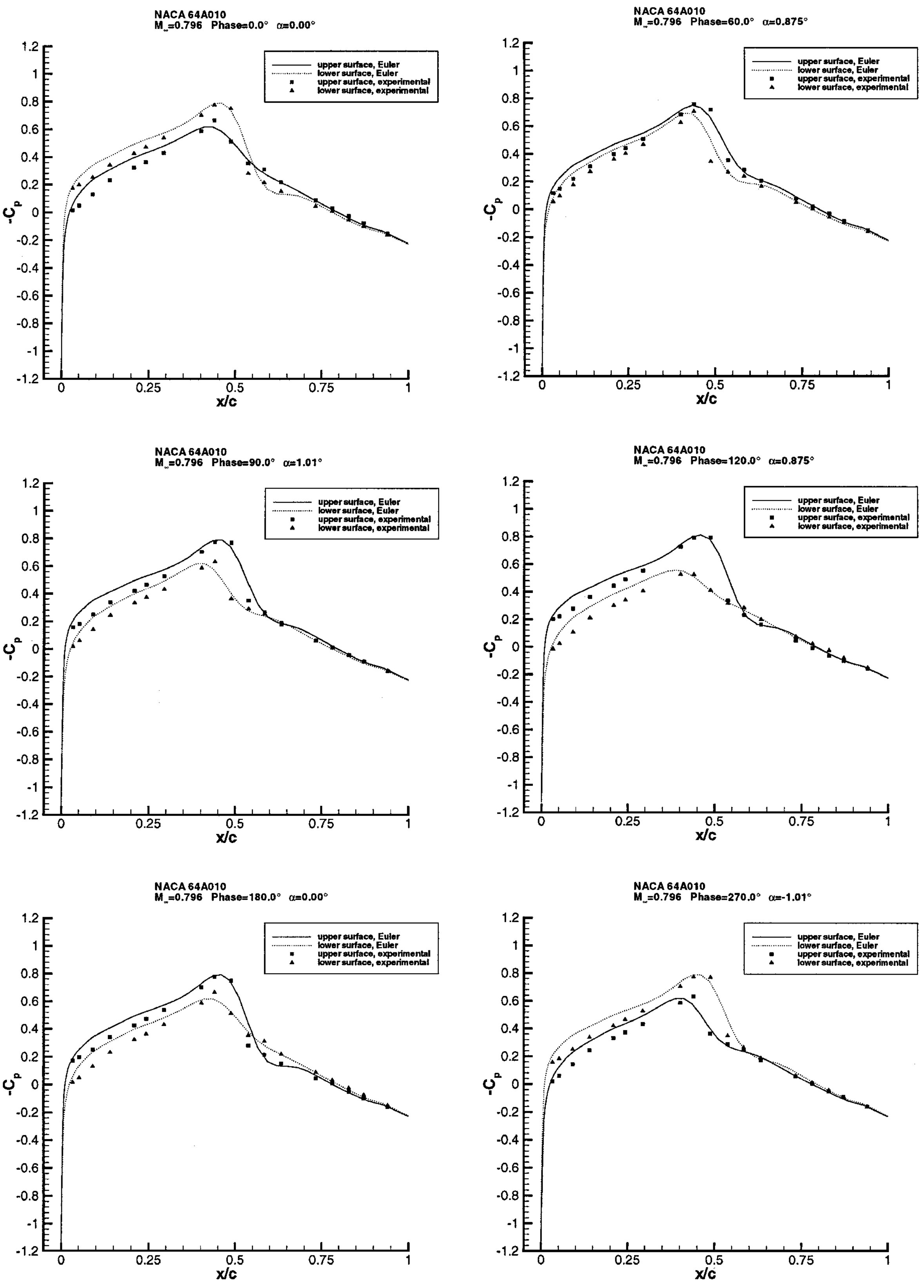

Fig. 11 Time history of surface pressure (x–y plane).

mesh with a simple O-grid topology. Figure 2 shows the closeup view of the original one-block mesh before deformation. An extreme pitch down angle of 45 deg is then applied to the airfoil. Figure 3 shows the regenerated mesh after the deformation. Good grid quality is retained even with such a drastic deformation. This is due to the feature of AIM3D that keeps points near the surface at the same angle with respect to the original grid.

Further testing of AIM3D was done on a eight-block version of the NACA0012 grid. This was done with a more realistic pitch angle of 20 deg. Figure 4 shows the block division of the mesh. A closeup view can be seen from Fig. 5. The remeshing is complete with no crossover of cells or gaps between blocks. Grid lines are also smooth across interior block to block boundaries.

AIM3D has been successfully tested on three-dimensional single- and multiblock structured grids. Figures 6 and 7 show some of the grids that were tested using AIM3D. Figure 6 shows the geometry and block boundaries of a wing–pylon mesh of 107,118 grid points in 14 blocks that was successfully deformed by pitching it up an

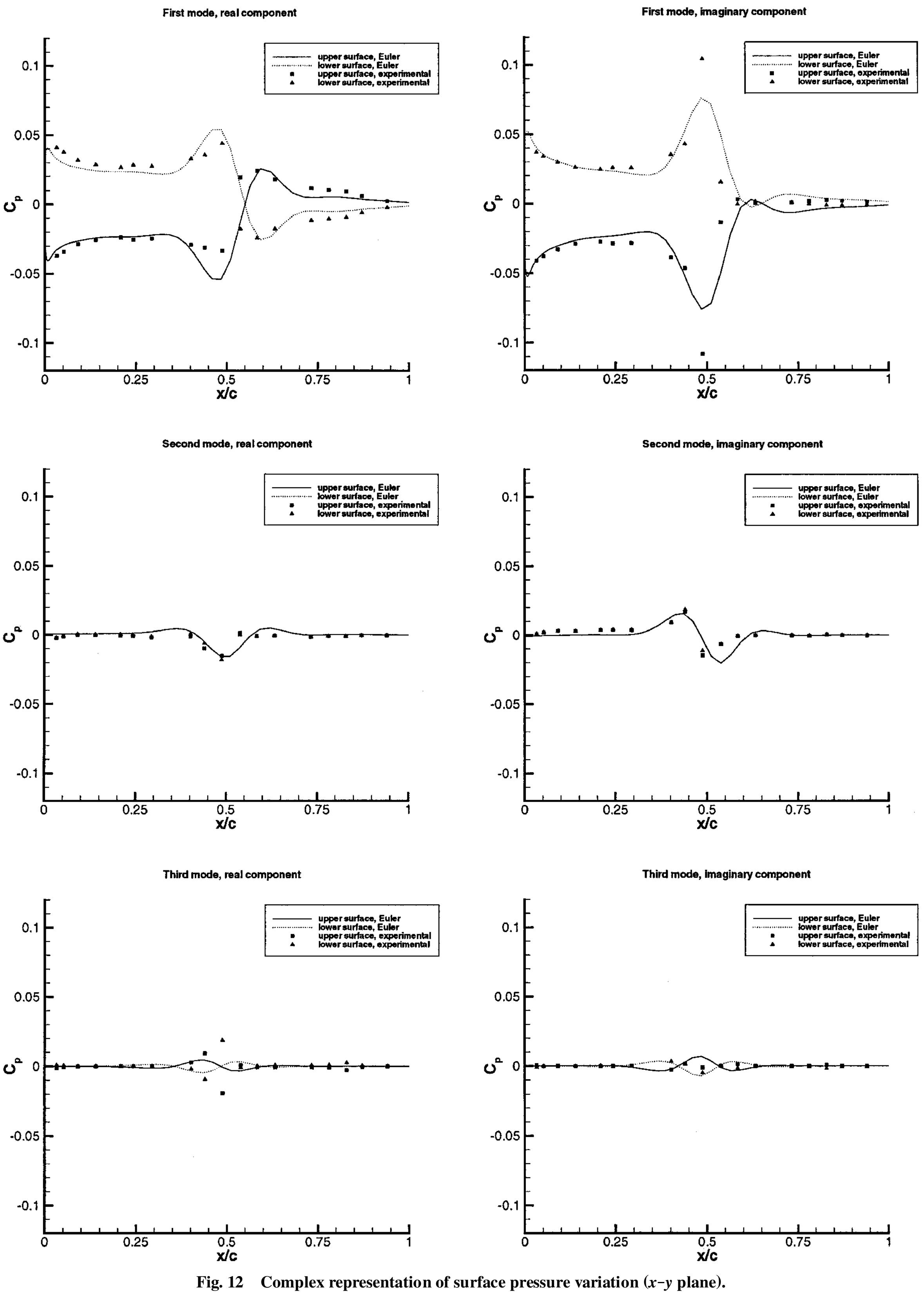

Fig. 12 Complex representation of surface pressure variation (x–y plane).

angle of 10 deg. Figure 7 shows a generic store (body plus fin) that was successfully rotated about its longitudinal axis by 10 deg. A mesh of 54,774 grid points in 22 blocks is used. Figures 8 and 9 show a closeup cross-sectional view of the store off-body meshes before and after the rotation, respectively.

Flow over an Oscillating Airfoil

Both the flow solver and the moving grid method are tested on the unsteady flow over an oscillating airfoil. Figures 10–12 show the comparison of computed and experimental data[24] for an NACA64A010 airfoil pitching in a freestream of Mach number 0.796. The computed history of the lift coefficient vs angle of attack is shown in Fig. 10 along with corresponding experimental data. Figure 11 shows the surface pressure distributions at different phase angles in a pitching period. Shock motion on the top and bottom surfaces of the airfoil are clearly seen. Figure 12 shows the first three Fourier modes of the surface pressure distributions of both the computation and the experiment. The nonzero higher harmonic components near the shock wave indicate there is significant nonlinear behavior due to the existence of the shock wave.

The flow solver has been run with both the general moving grid method presented in this paper and a grid with rigid-body motion. The results are nondistinguishable on the plots. The general moving grid method, however, is applicable to cases where there is arbitrary body deformation where a rigid-body grid is not applicable. Such is the case in the following example.

Flow over a Three-Dimensional Wing with Structural Deformation

The AGARD 445.6 wing[25,26] with structural deformation is used as a test case for the flow solver and the moving grid method. We have demonstrated the capability of the flow solver and the moving grid method in calculating three-dimensional unsteady flows within a short amount of time on a 32-processor parallel computer. Figure 13 shows the wing surface deformation for the first four structural deformation modes of the AGARD 445.6 wing. Figures 14 and 15 show a cross section of the multiblock grid near the tip of the wing for mode 2. The thick lines denote the block boundaries. It can be seen that the deformed grid maintains good quality near the wing surface and smooth transition between blocks without discontinuity even though the grid deformation and the flow calculations are performed independently on separate processors in the multiblock parallel calculations. The code has been used for coupled unsteady fluid/structure simulations of the 445.6 wing. In a typical run of the coupled simulation in Euler mode with 176,601 grid points in 32 blocks on a 32-processor networked personal computer cluster, the grid deformation part takes less than 1% of the total CPU time, which indicates that the moving grid method is very efficient. Detailed results and comparisons of the calculated flutter boundary with experimental data are presented in Ref. 21.

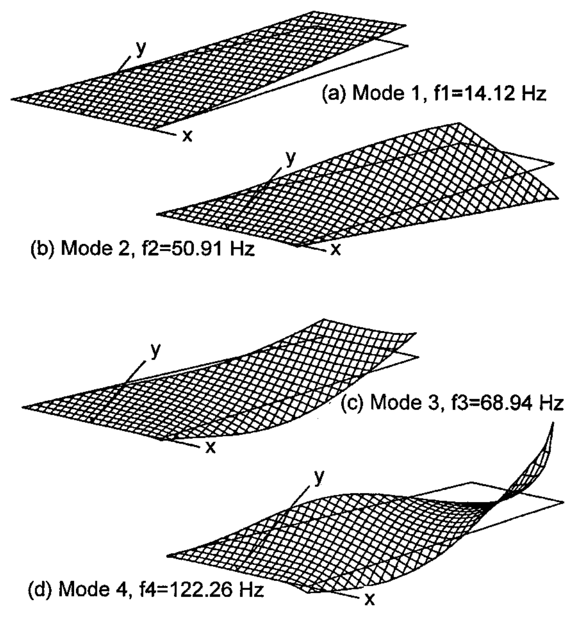

Fig. 13 Modal deflections for the 445.6 wing.

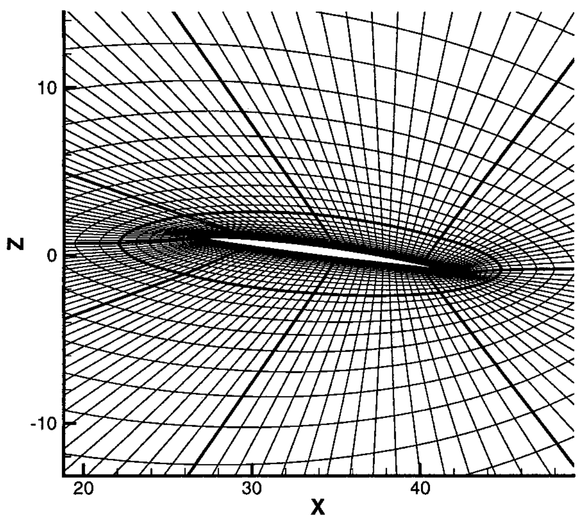

Fig. 14 Grid in a cross section of the AGARD 445.6 wing near the wing tip one instant during motion of the second vibrational mode.

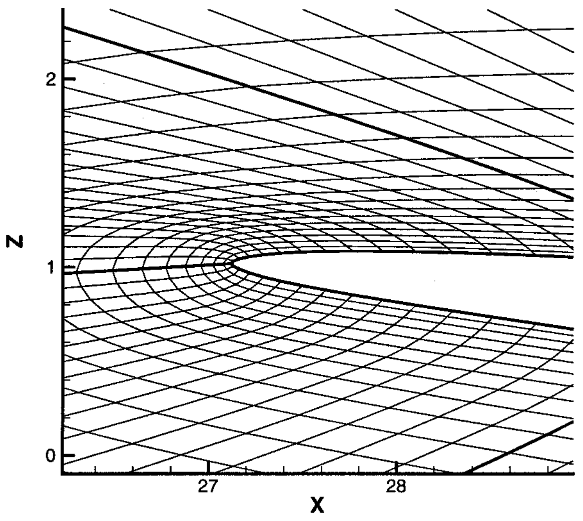

Fig. 15 Grid closeup in a cross section of the AGARD 445.6 wing near the wing tip one instant during motion of the second vibrational mode.

For Navier–Stokes calculations with very small grid spacing near the wing surface, grid crossing may occur when the surface deformation causes displacement larger than the grid spacing. This is a common difficulty with all existing codes to the authors' knowledge. In our implementation for Navier–Stokes calculations, the deforming grid method is modified to use an Euler grid as a base grid from which intermediate meshes are generated at each time step and then redistributed in the normal direction near solid walls to obtain the needed dynamic viscous grids. This strategy has been successfully used in unpublished work involving the 445.6 wing and also a complex aircraft configuration.

Conclusions

An automatic grid remeshing code was developed and tested for two- and three-dimensional Euler flow calculations. The code, AIM3D, accepts new surface deformations due to body motion and regenerates new off-body grids while maintaining the original grid quality. The code, which uses a mixture of algebraic and iterative methods, allows for mixed boundary conditions and is robust. More significantly, the method allows multiblocks of grids that can be distributed in an unstructured fashion to be used. The advantage of this methodology is that TFI can be used on each processor for the blocks that are allocated to the processor for parallel computation. Minimum communication between blocks and processors is needed, and the nature of the method ensures automatic matching of the edge and block-to-block faces provided the original grid block-to-block faces are matched.

Acknowledgments

This project was sponsored by the Defense Research Directorate, DSTA Singapore. The authors would like to thank N. H. Teng and J. K. Tan for their encouragement and support. Computations of this work has been performed on the AENEAS parallel personal computer cluster and also the SGI Origin 2000 computer at the University of California, Irvine.

References

[1]Lee-Rausch, E. M., and Batina, J. T., "Wing Flutter Boundary Prediction Using Unsteady Euler Aerodynamic Method," *Journal of Aircraft*, Vol. 32, No. 2, 1995, pp. 416–422.

[2]Lee-Rausch, E. M., and Batina, J. T., "Wing Flutter Computations Using an Aerodynamic Model Based on the Navier–Stokes Equations," *Journal of Aircraft*, Vol. 33, No. 6, 1996, pp. 1139–1147.

[3]Bhardwaj, M. K., Kapania, R. K., Reichenbach, E., and Guruswamy, G. P., "Computational Fluid Dynamics/Computational Structural Dynamics Interaction Methodology for Aircraft Wings," *AIAA Journal*, Vol. 36, No. 12, 1998, pp. 2179–2186.

[4]Byun, C., and Guruswamy, G. P., "Aeroelastic Computation on Wing–Body-Control Configurations on Parallel Computers," *Journal of Aircraft*, Vol. 35, No. 2, 1998, pp. 288–294.

[5]Ji, S., and Liu, F., "Flutter Computation of Turbomachinery Cascades Using a Parallel Unsteady Navier–Stokes Code," *AIAA Journal*, Vol. 37, No. 3, 1999, pp. 320–327.

[6]Prananta, B. B., and Hounjet, M. H. L., "Large Time Step Aero-Structural Coupling Procedures for Aeroelastic Simulation," *Proceedings of the International Forum on Aeroelasticity and Structural Dynamics*, Vol. 2, Confederation of European Aerospace Societies, Associazione Italiana di Aeronautica ed Astronautica, Rome, 1997, pp. 63–70.

[7]Chen, H. H., Chang, K. C., Tzong, T., and Cebeci, T., "Aeroelastic Analysis of Wing and Wing/Fuselage Configurations," AIAA Paper 98-0907, Jan. 1998.

[8]Reuther, J., Alonso, J., Rimlinger, M. J., and Jameson, A., "Aerodynamic Shape Optimization of Supersonic Aircraft Configurations via an Adjoint Formulation on Distributed Memory Parallel Computers," *Computers and Fluids*, Vol. 28, No. 4–5, 1999, pp. 675–700.

[9]Jones, W. T., and Samareh-Abolhassani, J., "A Grid Generation System for Multi-Disciplinary Design Optimization," *12th AIAA Computational Fluid Dynamics Conference Proceedings*, Pt. I, AIAA, Washington, DC, 1995, pp. 474–482.

[10]Batina, J. T., "Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes," *AIAA Journal*, Vol. 28, No. 8, 1990, pp. 1381–1388.

[11]Robinson, B. A., Batina, J. T., and Yang, H. T. Y., "Aeroelastic Analysis of Wings Using the Euler Equations with a Deforming Mesh," *Journal of Aircraft*, Vol. 28, No. 11, 1991, pp. 778–788.

[12]Tavella, D. A., "Lift of Delta Wings with Leading-Edge Blowing," *Journal of Aircraft*, Vol. 25, No. 6, 1988, pp. 522–524.

[13]Tsai, H. M., Dong, B., and Lee, K. M., "The Development and Validation of a Three-Dimensional Multiblock, Multigrid Flow Solver for External and Internal Flows," AIAA Paper 96-0171, 1996.

[14]Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge–Kutta Time-Stepping Schemes," AIAA Paper 81-1259, June 1981.

[15]Jameson, A., "Analysis and Design of Numerical Schemes for Gas Dynamics 1, Artificial Diffusion, Upwind Biasing, Limiters and Their Effect on Accuracy and Multigrid Convergence," *International Journal of Computational Fluid Dynamics*, Vol. 4, Nos. 3–4, 1995, pp. 171–218.

[16]Jameson, A., "Analysis and Design of Numerical Schemes for Gas Dynamics 2, Artificial Diffusion, Upwind Biasing, Limiters and Their Effect on Accuracy and Multigrid Convergence," *International Journal of Computational Fluid Dynamics*, Vol. 5, Nos. 1–2, 1995, pp. 1–38.

[17]Jameson, A., "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings," AIAA Paper 91-1596, June 1991.

[18]Alonso, J. J., and Jameson, A., "Fully-Implicit Time-Marching Aeroelastic Solutions," AIAA Paper 94-0056, Jan. 1994.

[19]Liu, F., and Ji, S., "Unsteady Flow Calculations with a Multigrid Navier–Stokes Method," *AIAA Journal*, Vol. 34, No. 8, 1996, pp. 2047–2053.

[20]Zhu, Y., "Time-Accurate Calculations of Unsteady Flows with Aeroelastic Applications," M.S. Thesis, Dept. of Mechanical and Aerospace Engineering, Univ. of California, Irvine, CA, Dec. 1998.

[21]Liu, F., Cai, J., Zhu, Y., Tsai, H. M., and Wong, A. S. F., "Calculation of Wing Flutter by a Coupled Fluid-Structure Method," *Journal of Aircraft*, Vol. 38, No. 2, 2001, pp. 334–342; also AIAA Paper 2000-0907, Jan. 2000.

[22]Soni, B. K., "Two- and Three-Dimensional Grid Generation for Internal Flow Applications of Computational Fluid Dynamics," AIAA Paper 85-1526, 1985.

[23]Kodama, Y., "The Development and Validation of a Three-Dimensional Multiblock, Multigrid Flow Solver for External and Internal Flows," AIAA Paper 96-0171, Jan. 1996.

[24]Davis, S. S., "NACA64A010 Oscillatory Pitching," *Compendium of Unsteady Aerodynamics Measurements*, Rept. 702, AGARD, 1982.

[25]Yates, E. C., "AGARD Standard Aeroelastic Configurations for Dynamic Response, I—Wing 445.6," NASA TM 100492, Aug. 1987.

[26]Yates, E. C., Land, N. S., and Foughner, J. T., "Measured and Calculated Subsonic and Transonic Flutter Characteristics of a 45° Sweptback Wing Planform in Air and in Freon-12 in the Langley Transonic Dynamics Tunnel," NASA TN D-1616, March 1963.

E. Livne
Associate Editor