JOURNAL OF AIRCRAFT
Vol. 38, No. 2, March–April 2001

Numerical Simulation of Dynamically Deforming Aircraft Configurations Using Overset Grids

Yuval Levy*
Technion—Israel Institute of Technology, 32000 Haifa, Israel

An automatic scheme for the simulation of the flow about a deforming aircraft configuration is presented. The scheme utilizes the overset grid method to perform elastic deformations and to vary the elevator deflection angle, thus avoiding the regeneration of the computational mesh. As a part of the scheme, a new method for automatic collar grid generation is developed and presented. The method is especially efficient for sharp intersections. The generated collar grids are smooth and orthogonal and clustered toward the walls of the intersecting geometries. The algorithm is incorporated in a flow solver that utilizes the overset grid approach. An example is given for the case of the flow about a maneuvering, elastic, fuselage-wing-tail configuration. The changes to the configuration in the course of the computation require a nonfailing automatic grid-generation procedure because of the changes to the fuselage-wing and the fuselage-tail intersection regions. The results show good agreement with computational results obtained using the patched approach.

Introduction

COMPUTATIONAL-FLUID-DYNAMICS (CFD) methods have advanced to the point where numerical flow simulations about complex geometries are widely used for research and design purposes.[1,2] However, the use of CFD methods for design purposes is still limited because of the high costs of numerical simulations. During the design process, the shape of components and their intersection with neighboring components are changed. Each change requires the generation of a new computational mesh and that the simulation would be started from scratch.

The grid-generation process in itself provides one of the main obstacles in performing numerical simulations. Using a structured mesh, there are two different approaches to mesh generation. The first is the patched grid approach and the second is the overset grid approach, known as chimera.[3]

In the patched grid approach the physical domain is divided into zones that touch each other, and the computational mesh is generated for each zone separately. A small overlap between neighboring zones allows the smooth transfer of information between zones. This approach allows the generation of a computational mesh for complex geometries. However, in a case of a geometry change, the computational mesh associated with the component that was changed and the neighboring meshes must be adjusted or regenerated. In the case of minor changes to the geometry, adjustments to the original mesh provide a satisfactory solution, whereas regeneration is required for major geometry changes.

In the chimera grid approach[3,4] a separate computational mesh is generated for each component, such as the fuselage or the wing of an aircraft, separately. An outer mesh is generated so that it fully includes the meshes of all of the components. The domain decomposition in this approach is simplified, and the number of zones that are required is substantially lower. However, the generation of grids surrounding each component is performed independently, and points of a certain mesh may be located within the solid boundaries of the aircraft. Regions located within solid boundaries are called "holes." Grid points that are in the holes are treated by excluding them from the solution process and by using interpolations to update the edges of the holes. The calculations to determine the holes and hole boundaries require a considerable computational effort that, in a moving body simulations, has to be added to the overall computational cost.

The chimera approach provides an elegant solution to generating meshes surrounding complex geometries. However, difficulties arise around intersecting components, e.g., fuselage and a wing, as they create overlapping holes. Parks et al.[5] offered a solution by creating "collar grids" to provide appropriate interpolation stencils around the intersection region. Chan and Buning[4] presented a method for generating collar grids using hyperbolic grid-generation methods. They use the intersection line as an initial curve to generate the surface of the collar grid by marching away from the curve on both sides. Once the surface is generated, the collar grid is generated by marching away from the surface. One coordinate of the collar grid runs along the body surface from the wing, through the wing-fuselage intersection point, and then onto the fuselage. A second coordinate is normal to the wing and body surface at all points along the body surface. Thus, one slice of the collar grid is coincident with the wing and fuselage.

The method presented by Chan and Buning is versatile and allows the generation of collar grids for general geometries. However, in the case of a sharp intersection, e.g., a wing-fuselage intersection, the surface of the collar grid is not smooth, and therefore the collar grid itself is not smooth around the intersection line. Moreover, the resolution of the collar grid in the vicinity of the intersection line is limited, the grid lines are largely skewed, and the grid-generation process may fail if the resolution near the intersection is too high. Raising the resolution in the vicinity of the intersection can result in a failure of the grid-generation process because grid lines start to cross each other. This tendency can be clearly seen by examining the following example.

Figure 1 shows the surface of a collar grid for a fuselage-wing intersection. This surface is used to generate the collar grid. The figure also contains a slice of the grid. It is important to note that the surface of the collar grid is approximately the same for any grid-generation technique because it is based on the surfaces of the intersecting components. As will be shown, the main differences are in the distribution of the internal grid points of the collar grid. Figure 2 shows a section of the collar grid on the top side of the wing. The clustering around the intersection line is limited because grid lines that emanate from that region tend to cross each other when using the hyperbolic solver. This can be seen in the figure by the dark region, just above the intersection line. A close-up of the same slice is presented in Fig. 3. The cells closest to the intersection line are largely skewed, and the grid lines are not smooth. A sharp direction change is apparent just above the intersection line. This sharp change results in a large error in the approximation of flow variable derivatives. Because this

Received 29 October 1999; revision received 7 November 2000; accepted for publication 10 November 2000.

*Senior Lecturer, Faculty of Aerospace Engineering.

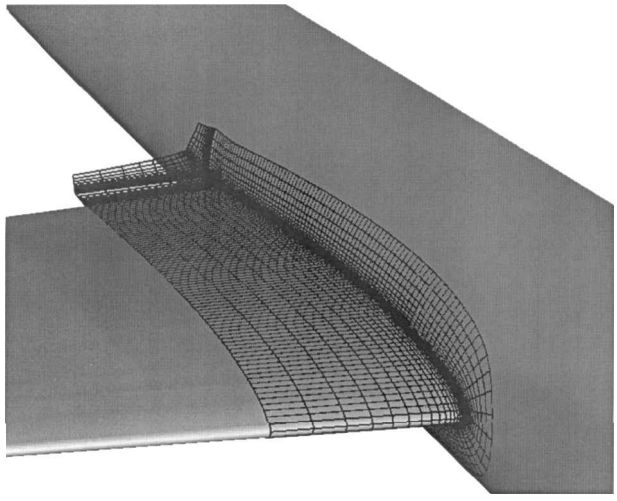

Fig. 1 Surface of a collar grid for a fuselage-wing intersection as generated with a hyperbolic grid generator.

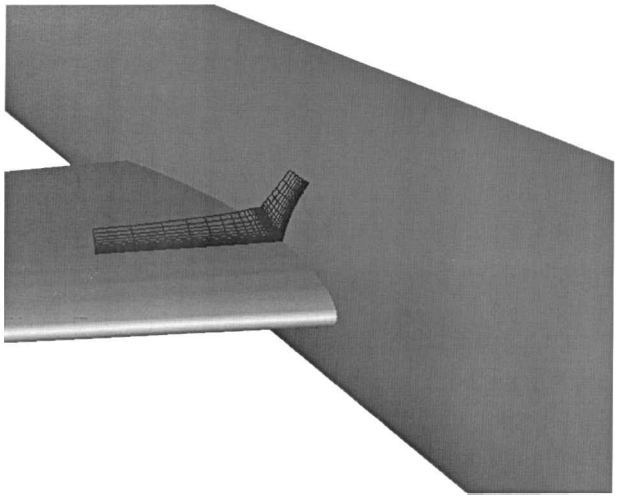

Fig. 2 Slice of a hyperbolically generated collar grid.

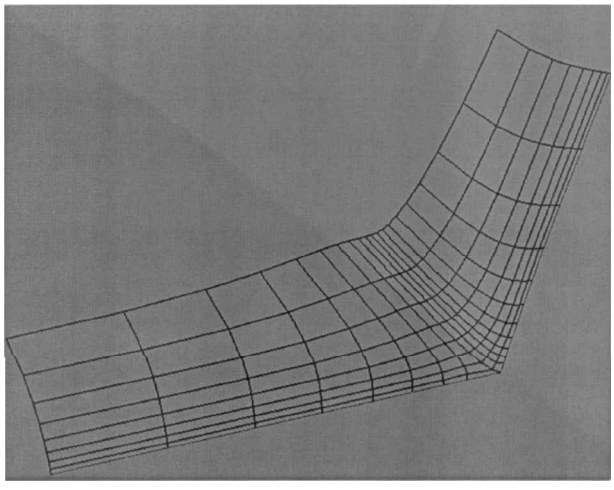

Fig. 3 Close-up of a slice of a hyperbolically generated collar grid.

region experiences large gradients, the errors hinder the accuracy of the solution in the intersection area, which, in turn, affects the whole solution.

The need to automatically generate collar grids arises when the configuration in question experiences a geometry change, for example, a computational scheme that determines the trimmed configuration of an aircraft by changing its incidence angle and its elevator deflection angle.[6,7] Raveh and coworkers[6,7] used the patched grid approach and used blending for changing the elevator deflection angle. This results in a lower elevator efficiency compared to a rigid-body rotation of the elevator. The patched grid approach was also used by Byun and Guruswamy[8] for aeroelastic applications on a full aircraft. However, they solved the flow about a wing only and therefore did not deal with intersecting bodies.

To facilitate the geometry changes, a new algorithm for automatic collar grid generation is developed and presented. The algorithm permits the generation of a smooth collar grid with high resolution in sharp intersection regions without the risk of failure during the calculation of the grid points. The resulted collar grids allow for changes in the wing-fuselage intersection caused by elastic deformations. The collar grids also account for changes in the tail-fuselage intersections caused by elevator deflections.

The collar grid-generation algorithm is presented in the framework of a flow solver (EZNSS) that was modified to accommodate the special collar grid. The flow solver is capable of solving either the inviscid Euler or the full Navier–Stokes equations, using one of two implicit algorithms. The first is the Beam and Warming algorithm,[9] and the second is the flux-splitting algorithm reported by Steger et al.[10] The EZNSS code also has the capability to perform elastic correction to the model, based on static aeroelasticity and maneuver trim corrections. The whole code is parallelized, and the flow simulations were conducted on a Silicon Graphics Origin 2000.

In this paper the flow about a maneuvering, elastic, fuselage-wing-tail configuration is simulated by using the EZNSS capabilities. The results are compared with results obtained using a different code (FA3DMB), a finite volume code that utilizes the patched grid approach. The results show that using the chimera grid approach for geometry changes allows the simulation of the flow around changing intersections more easily and that the efficiency of the elevator is more accurate.

Single Mesh Topology

To minimize the input parameters that a user must set before running a simulation, the types of separate computational meshes are limited to three types. The first is a regular type mesh (Cartesian), which is primarily used for describing the walls of a wind tunnel; the second is a C-H grid topology, which is used primarily for wings and pylons; and the third is a C-O or an O-O grid topology for outer meshes or for slender bodies such as a fuselage. The use of a C-H grid topology for wings provides the means to simulate the flow around the wing tip with high resolution.

Generation of the grid surrounding each component of the geometry is performed separately. Using a commercial grid-generation package,[11] the surface grid is created based on the surface definition. Grid points are distributed so that regions containing important flow features are sufficiently resolved. These include regions where shocks are expected, regions near intersections between components, and other areas where large flow gradients are expected. Once the surface grid is defined, the interior grid points are calculated using hyperbolic or elliptic methods. In the case of a hyperbolic grid generator, the grid is generated based on the surface grid by marching away from the surface, and the outer boundary is formed as a part of the process. In the case of an elliptic grid generator, the outer boundary has to be defined prior to the grid-generation process.

The flow solver is modified to include a set of routines that detect the type of mesh and set the appropriate boundary conditions. The flow solver also includes a suite of routines to calculate inter-grid communications. These routines are invoked every time the configuration experiences a geometry change. An important part of the procedure is the automatic creation of the collar grids. The user defines the intersecting elements of the configuration, and the collar grid is generated by the procedure described in the following section.

Collar Grid Generation

The collar grids are based on the C grid topology. It is a natural choice because this type of collar grid is used to provide the link between the C grid of the wing (or empennage component) and the

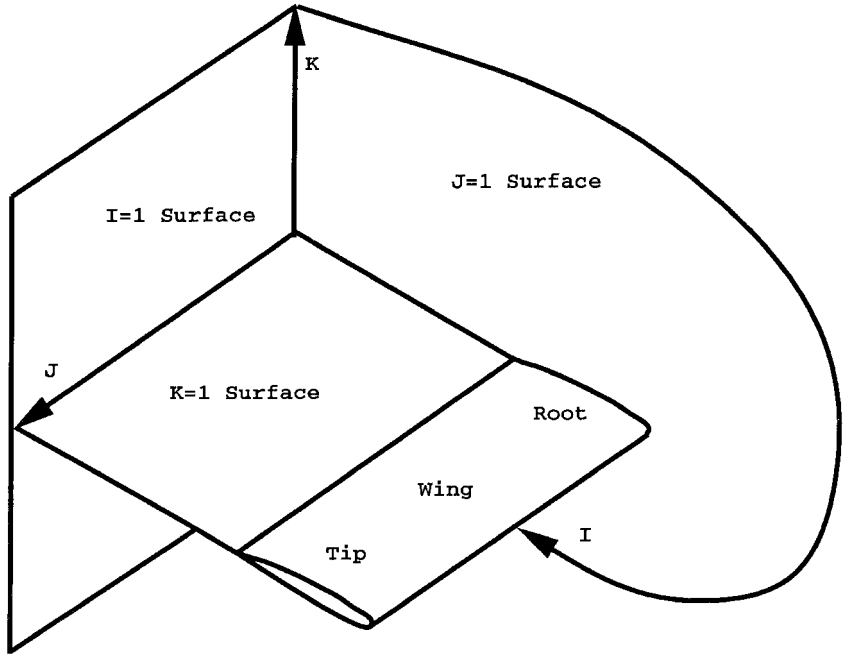

Fig. 4 C-grid topology.

fuselage grid. Consequently, two of the boundaries of the collar grid are surfaces of the intersecting geometries: one of the fuselage and the other of the wing. The remaining four boundaries are within the computational domains surrounding the fuselage and the wing. In contrast to the hyperbolically generated collar grids, one coordinate originates at the fuselage surface and runs outward along the wing surface. A second coordinate originates at the wing surface and runs along the fuselage surface.

The grid-generation process is comprised of three steps. The first is the surfaces and boundaries generation, the second is an algebraic distribution, and the third is smoothing using an elliptic scheme.

Fuselage Surface

The surface tangent to the fuselage is generated by extending the wing mesh one grid point into the fuselage. The intersection points of the wing grid lines and the surface of the fuselage are calculated along the wing-fuselage intersection line and an additional line above it. These lines are defined as follows: all of the lines in the J direction (see Fig. 4), where $K = 1$, and I starts before the upper trailing-edge point and ends after the lower trailing-edge point.

This process is repeated for a prescribed value of K, which would ensure that the collar grid completely surrounds the holes on the surface of the fuselage. Straight lines are generated based on the pairs of points with the same I and J values, and a prescribed number of points are distributed along each line. At this stage all of them are inside the fuselage. The points are then projected onto the surface of the fuselage. The projection process is similar to finding the intersection between a line and a surface and is described next. By choosing an exponential distribution function, the points on the defined surface are clustered toward the wing-fuselage intersection.

Intersection Between a Line and a Discrete Surface

As was just mentioned, the fuselage surface is comprised of a series of grid points that are the intersections between grid lines of the wing mesh and the surface of the fuselage mesh. The intersection point can be easily found assuming both line and surface are known analytical functions.

Let $P_1 = \{x_1, y_1, z_1\}$ and $P_2 = \{x_2, y_2, z_2\}$ be the points that define a line and $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ be the position associated with P_1 and P_2 relative to zero, then the line passing through the points is given by

$$\boldsymbol{V}_l = t\boldsymbol{V}_1 + (1-t)\boldsymbol{V}_2 \tag{1}$$

where $-\infty < t < \infty$. Let $f(x, y, z) = 0$ be the definition of a surface, then the intersection between the line $\boldsymbol{V}$ and the surface f is given by $f(\boldsymbol{V}) = 0$. Depending on the function f, the intersection point can be found analytically or numerically.

In our case the fuselage is given by discrete points, and therefore the intersection point must be found numerically. An efficient method to locate the intersection point is based on a bilinear interpolation within a cell of the fuselage mesh. Let P_3, P_4, P_5, and P_6 be the points that define a surface cell and V_3, V_4, V_5, and V_6 be their respective positions relative to zero, then a point in the cell is given by

$$\boldsymbol{V}_c = s[r\boldsymbol{V}_3 + (1-r)\boldsymbol{V}_4] + (1-s)[r\boldsymbol{V}_5 + (1-r)\boldsymbol{V}_6] \tag{2}$$

where $0 < r, s < 1$.

If a line intersects a certain cell, then $V_l = V_c$ and $0 < r, s < 1$. The parameters r, s, and t can be found by the solution of the 3×3 system given by Eqs. (1) and (2). If $0 < s < 1$, the intersection point is between the points P_1 and P_2 and the intersection point is found by a linear interpolation; otherwise, the point is found through extrapolation. If r or s are not between zero and one, then the line does not intersect the particular cell. The values of r and s are then used to move to the next cell, and the system is solved until the intersection point is found.

Wing Surface

The surface tangent to the wing is generated following a similar process to that used for generating the fuselage surface. The points that define the wing-fuselage intersection are used together with points on the wing surface to generate the lines that are inside or outside the wing, depending on the wing's curvature. Points are exponentially distributed on these lines and then projected onto the wing surface. The surface is also extended in the $\pm I$ direction to cover the fuselage hole.

Volume Boundaries

The other four boundaries of the collar grid are generated based on the surface boundaries. The edges of two of the surfaces are normal to the fuselage and the wing surface, respectively, and the other two are set so that they smoothly close the volume covered by the collar grid. It is achieved by using the same projection algorithms used for the surface boundary generation.

Grid Generation

Once the boundaries are defined, internal points are computed through interpolations based on the boundary values. The points are smoothed using an elliptic grid generator based on the Thompson, Thames, and Mastin algorithm.[12] The smoothing procedure is applied to $I =$ constant slices to ensure a smooth orthogonal mesh. Using the TTM algorithm guarantees that in each I slice the coordinates are orthogonal and smooth. Control functions are used to ensure a point distribution similar to that of the boundaries. The whole process requires a negligible amount of computer time because only a few iterations are needed for convergence.

The resulting collar grid is smooth, and it maintains orthogonality everywhere it is defined even in the region close to the intersection line. In addition, its points are clustered toward the wing and the fuselage surfaces in a way that allows high resolution of the intersection region.

Application

The flow solver was tested on a fuselage-wing-tail configuration where the tail is an all-movable tail that acts as the elevator. Figure 5 shows the model's complete geometry and the computational mesh of the main components (every other grid point in each direction is shown). The fuselage grid is based on an O-O type mesh. C-H type grid topology is used for the wing and the tail. A similar configuration was used by Raveh and coworkers[6,7] for a simulation of a maneuvering elastic aircraft. The simulation includes three levels of iterative processes. The innermost level includes the iterations of the flow solver. The next level, performed every prescribed number of flow solver steps, introduces elastic corrections. The outer level contains maneuver-trim corrections and is applied every prescribed number of elastic corrections. A detailed description of the trim algorithm can be found in Refs. 6 and 7.

Figure 6 shows the surface of the collar grid for the wing-fuselage intersection as generated by the newly proposed method. Also shown is one slice of the grid. In fact, three of the boundaries of the collar grid are presented in the figure: two on the surface and the slice for $I = 1$. As seen from the figure, the surface is similar to that of the collar grid generated by the hyperbolic generator (see Fig. 1).

Figure 7 shows a slice of the collar grid. The grid lines are smooth, orthogonal, and clustered near the surfaces of the fuselage and the wing. In contrast to the hyperbolic grid, where the grid lines normal to the surface are of limited length, the collar grid is extended well into the computational domains defined by meshes of the fuselage and the wing. A close-up of the same slice is shown in Fig. 8.

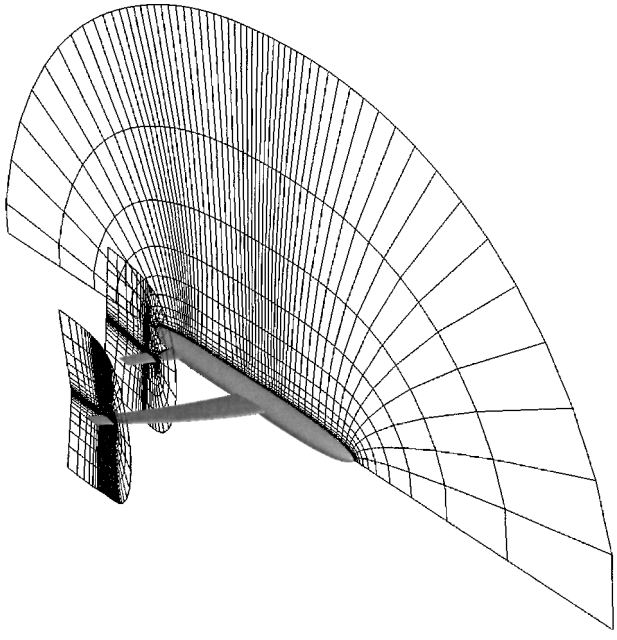

Fig. 5 Overall view of the geometry and the grid system.

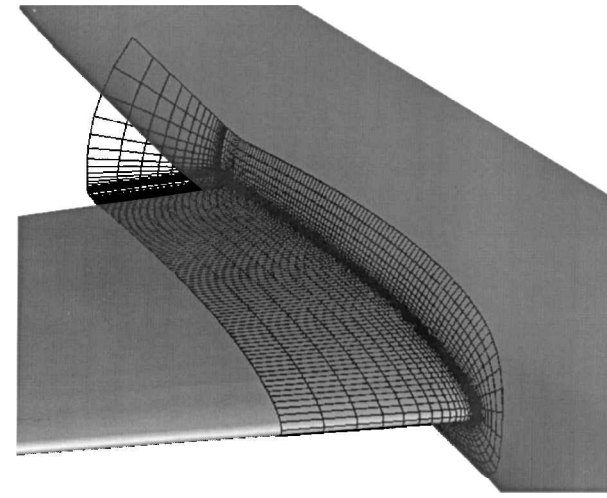

Fig. 6 Collar grid for a fuselage-wing intersection as generated by the proposed method.

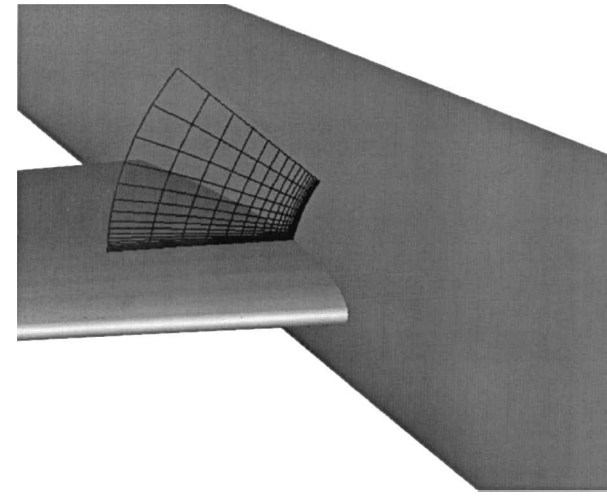

Fig. 7 Slice of the fuselage-wing collar grid.

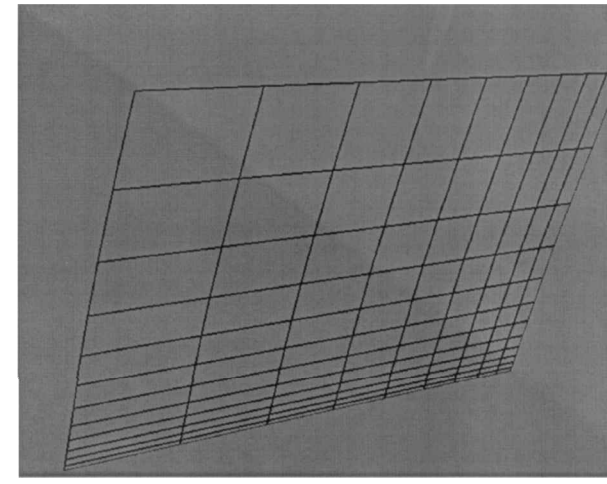

Fig. 8 Close-up of a slice of the fuselage-wing collar grid.

The cells in the intersection regions are practically rectangular and smooth.

Raveh and coworkers[6,7] utilized a code that used the patched grid approach, and therefore every change to the boundaries of a certain patch affected the neighboring patches. The patched grid approach can handle small deformation, such as the grid changes caused by elastic corrections, with relative ease. However, in the case of large geometry changes, such as elevator deflections as required for trim

corrections, the grids of the elevator and the surrounding patches have to be regenerated. Raveh and coworkers[6,7] have solved the problem by the use of blending. The elevator deflection ranges over a few grid lines that are moved by a fraction of the total elevator deflection, such that the root remains at its original position and the last point at the blended region reaches the total deflection. The use of blending provides a reasonable approximation to the new geometry everywhere except for the fuselage-tail intersection region. The approximation results in a lower computed efficiency of the elevator, which introduces an error in the computation.

Using a flow solver that utilizes an overset grid method allows the performance of large geometry changes because changes to the grid of a certain component do not affect the grids of neighboring components. For example, a change of the elevator angle is achieved by a simple rotation of the elevator. Following the rotation, the intergrid communications are recalculated, and the collar grid is regenerated. This is achieved by including the necessary routines in the flow solver.

Results

Numerical flow simulations were conducted to verify that the flow solver, the collar grid generator, and the intergrid communication procedures are capable of handling a problem that includes geometry changes. A 3-g pull-up maneuver of a flexible aircraft at Mach 0.85 and height of 11,000 meters was chosen because a similar simulation was conducted by Raveh and coworkers[6,7] using a finite volume code that uses the patched grid approach (FA3DMB).[13] For a half-aircraft weight of 5,777 kg, a 3-g pull-up maneuver translates to lift coefficient of $C_L = 0.84$ and zero moment coefficient $C_M = 0$. The simulation started from initial guesses for the angle of attack and the elevator deflection angle. In the current case the initial angle of attack was $\alpha = 6.9$ deg, and the elevator deflection angle was $\delta = -10.6$ deg. Both values were obtained from trim analysis conducted with linear aerodynamics.

Figure 9 (taken from Ref. 14) shows the convergence history of the lift and moment coefficients. Both coefficients converge to their required values, and the convergence history follows a similar behavior to the FA3DMB simulations.[14] Figure 10 (taken from Ref. 14) shows the convergence history of the simulation. Also shown in the figure is the convergence history of a simulation of a rigid geometry. Large residual increases follow each geometry change. They arise as a result of the chimera overset method. Every control-surface deflection or elastic correction can change grid points in the fuselage mesh from being holes to become regular points. Because such points are far from convergence, their inclusion in the simulation contributes large values to the residuals. However, the number of these points is limited, and both implicit algorithms implemented in the code have fast convergence qualities. Therefore, the residual values decrease quickly to the values of a rigid-shape simulation.

Table 1 presents the maneuver trim results obtained using the EZNSS code with those obtained using the FA3DMB code. The small differences are acceptable given the fact the computer codes use entirely different algorithms and concepts. However, some of the differences can be attributed to the lower actual efficiency of the elevator when using blending (as done in the FA3DMB code) to change the deflection angle.

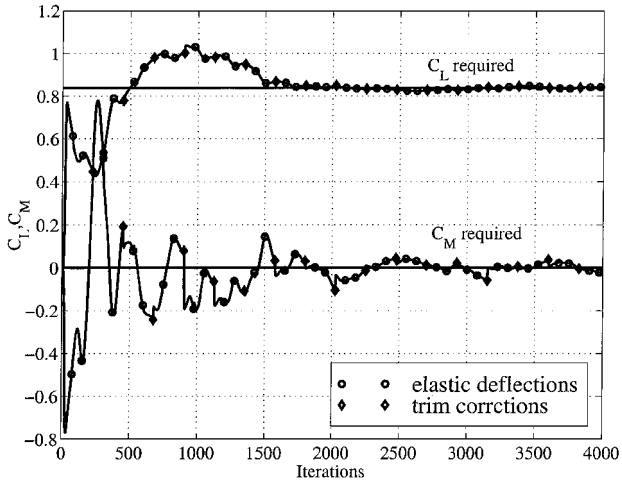

Fig. 9 C_L and C_M convergence history.

Table 1 Maneuver trim results

Parameter	EZNSS	FA3DMB
Angle of attack, deg	6.0	6.0
Elevator deflection, deg	−7.0	−7.4
Wing tip l.e. deflection, meter	0.91	0.82
Wing-tip wash-out angle, deg	2.8	2.4

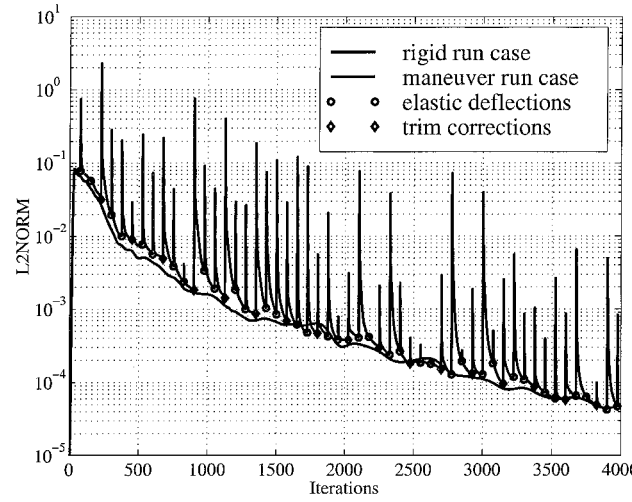

Fig. 10 Residual decay history.

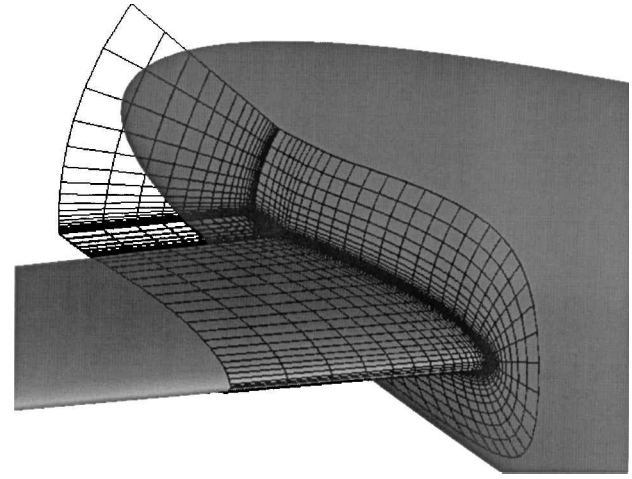

Fig. 11 Collar grid for the fuselage-tail intersection.

Figure 11 shows the walls of the fuselage-tail intersection collar grid and an additional slice of the grid. The stage that is presented in the figure is at the end of the calculation, where the deflection angle is −7.0 deg. The automatic algorithm was automatically invoked every time a trim correction was made to create the appropriate collar grid throughout the convergence path.

Conclusions

The results presented in this paper demonstrate the capabilities of the EZNSS flow solver. The code incorporates numerical simulations of the Navier–Stokes equations with static aeroelastic deformation and trim correction, using the chimera overset grid topology. To accommodate the geometry changes near the intersections, an automatic collar grid generation was developed. The choice of the coordinate topology for the collar grids was chosen to fit near

orthogonal intersection such as wing-fuselage intersections. The automatic grid-generation procedure facilitates the generation of collar grids around changing geometries and therefore allows the accurate simulation of the flow in these regions without the need to repeat the whole simulation.

The collar grids that are generated using the algorithm are smooth around the intersection region, and the grid lines are also orthogonal throughout the grid. Incorporating the algorithm with the flow solver allows the application of numerical schemes that require trim by using the chimera grid approach. Elevator deflection angle is applied by a rigid-body rotation of the elevator. Thus, a higher efficiency of the elevator is achieved, and the flow around the complete configuration is simulated with higher accuracy.

The automatic algorithm also provides the framework for an unsteady flow simulation of major geometry changes, for example, the time-accurate simulation of the flow about an elevator during a change in the deflection angle. Because the size of the collar grid is fixed, the calculation of flow in the next time step can be based on the preceding time step where a Newton iteration is added for convergence to the unsteady solution.

Acknowledgment

The author would like to thank D. E. Raveh from Technion for useful discussions during the course of this study.

References

[1] Agarwal, R., "Computational Fluid Dynamics of Whole-Body Aircraft," *Annual Review of Fluid Mechanics*, Vol. 31, 1999, pp. 125–169.

[2] Palmer, G., Buning, P., Yanowitz, H., and Venkatapathy, E., "Three-Dimensional Computational Analysis of Complex Launch Vehicle Configurations," *Journal of Spacecraft and Rockets*, Vol. 33, No. 1, 1996, pp. 49–53.

[3] Benek, J. A., Buning, P. G., and Steger, J. L.,"A 3-D Chimera Grid Embedding Technique," AIAA-85-1523, July 1985.

[4] Chan, W. M., and Buning, P. G.,"Surface Grid Generation Methods for Overset Grids," *Computers and Fluids*, Vol. 24, No. 5, 1995, pp. 509–522.

[5] Parks, S. J., Buning, P. G., Chan, W. M., and Steger, J. L., "Collar Grids for Intersecting Geometric Components Within the Chimera Overlapped Grid Scheme," AIAA Paper 91-1587, June 1991.

[6] Raveh, D. E., Karpel, M., and Yaniv, S., "Non-Linear Design Loads for Maneuvering Elastic Aircraft," *Proceedings of the 38th Israel Annual Conference on Aeroepace Sciences*, 1998, pp. 150–160.

[7] Raveh, D. E., and Karpel, M., "Structural Optimization of Flight Vehicles with Non-Linear Aerodynamic Loads," *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998.

[8] Byun, C., and Guruswamy, G. P., "A Parallel Multi-Block Moving Grid Method for Aeroelastic Applications on Full Aircraft," *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998, pp. 570–580.

[9] Beam, R. M., and Warming, R. F., "An Implicit Factored Scheme for the Compressible Navier–Stokes Equations," *AIAA Journal*, Vol. 16, No. 4, 1978, pp. 393–402.

[10] Steger, J. L., Ying, S. X., and Schiff, L. B., "A Partially Flux-Split Algorithm for Numerical Simulation of Unsteady Viscous Flows," *Workshop on Computational Fluid Dynamics*, Univ. of California, Davis, 1986.

[11] Steinbrenner, J. P., and Chawner, J. R., "Gridgen's Synergistic Implementation of Cad and Grid Geometry Modeling," *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Field Simulations*, Vol. 1, 1996, pp. 363–372.

[12] Thompson, J. F., Thames, F. C., and Mastin, C. W., "Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate Systems for Fields Containing Any Number of Arbitrary Two-Dimensional Bodies," *Journal of Computational Physics*, Vol. 15, No. 3, 1974, pp. 299–319.

[13] Yaniv, S., "Navier–Stokes Calculations for Rotating Configurations: Implementation for Rockets," *Journal of Spacecraft and Rockets*, Vol. 33, No. 5. 1996, pp. 756–758.

[14] Raveh, D. E., Karpel, M., and Levy, Y., "Integration of Structural Optimization Schemes with Computational Aerodynamics," *Proceedings of the 39th Israel Annual Conference on Aeroepace Sciences*, 1999, pp. 245–258.