OMI Overset Methods, Inc.
A Nonprofit Public Benefit Corporation

OMI 04-92-3

FINAL REPORT
for NASA GRANT 2-783
SUBMITTED TO

NASA Ames Research Center
Advanced Design Cycle Branch

Point of Contact:
Dr. Pieter G. Buning

By

Overset Methods, Inc.
262 Marich Way
Los Altos, CA 94022

On Automating Domain Connectivity for Overset Grids

Principal Investigators: Dr. Ing-Tsau Chiu / Dr. Robert L. Meakin

August 31, 1995

On Automating Domain Connectivity for Overset Grids

Ing-Tsau Chiu†
Robert L. Meakin‡
Overset Methods, Inc.
NASA-Ames Research Center
Mail Stop T227-2
Moffett Field, California 94035

ABSTRACT

An alternative method for domain connectivity among systems of overset grids is presented. Reference uniform Cartesian systems of points are used to achieve highly efficient domain connectivity, and form the basis for a future fully automated system. The Cartesian systems are used to approximate body surfaces and to map the computational space of component grids. By exploiting the characteristics of Cartesian systems, Chimera type hole-cutting and identification of donor elements for intergrid boundary points can be carried out very efficiently. The method is tested for a range of geometrically complex multiple-body overset grid systems. A dynamic hole expansion/contraction algorithm is also implemented to obtain optimum domain connectivity; however, it is tested only for geometry of generic shapes.

INTRODUCTION

The ability to accurately simulate unsteady flowfields about geometrically complex and moving component configurations is becoming increasingly important in the analysis of modern aircraft and launch vehicles. Although significant progress has been made in recent years to apply higher order computational methods to this class of problems, there are still obstacles which prevent computational fluid dynamics from making more of a direct impact on the design process. Currently available software for unsteady multiple body aerodynamics is very complex, and requires a large amount of human interaction and expertise. This, combined with limited computational capacity, greatly restricts the degree to which such problems can be studied. The primary objective of this research is to achieve algorithmic improvements which not only reduce computational demands associated with unsteady multiple body aerodynamics, but significantly reduce the corresponding demands on human resources.

Currently, the only viable high-order method of prediction for these problems is the so called Chimera[1], or overset grid approach. The approach involves the decomposition of problem geometry into a number of geometrically simple overlapping component grids. Grid components associated with moving bodies move with the bodies without stretching or distorting the grid system. The structure of individual grid components facilitates viscous boundary layer resolution, the use of implicit time-integration algorithms, and efficient use of memory resources. Also, the approach offers natural coarse-grained levels of parallelism that facilitate computation within distributed computing environments.

The focus of the present work is on automation of domain connectivity for systems of overset grids. The task of domain connectivity is a problem in computational geometry. Given a set of points identified as "data receivers" or Inter Grid Boundary Points (IGBPs), the task is to find suitable "donor elements" for each IGBP in the set. The problem is complicated by the fact that the set of IGBPs may change dynamically in response to body movement, which may be a prescribed motion, or a motion driven by aerodynamic and/or applied forces. There are two primary algorithmic tasks associated with this problem: identification of the set of IGBPs and identification of the corresponding set of donor elements. The following pages describe novel techniques that are being developed to address both of these tasks in a way that leads to automation of the entire process.

METHOD

Chimera

In a Chimera style overset grid approach, domain connectivity is achieved through interpolation of needed intergrid boundary information from solutions in neighboring grid components. Consider, for example, the simple two grid discretization of the airfoil shown in Figure 1. The problem domain is decomposed into a body-fitted grid system near the airfoil surface and a background Cartesian grid system which extends out to the far-field boundaries. The Cartesian grid completely overlaps the airfoil grid. Clearly, the airfoil grid outer boundary conditions can be interpolated from a solution in

---

†. Staff Scientist, Member AIAA
‡. Staff Scientist, Member AIAA

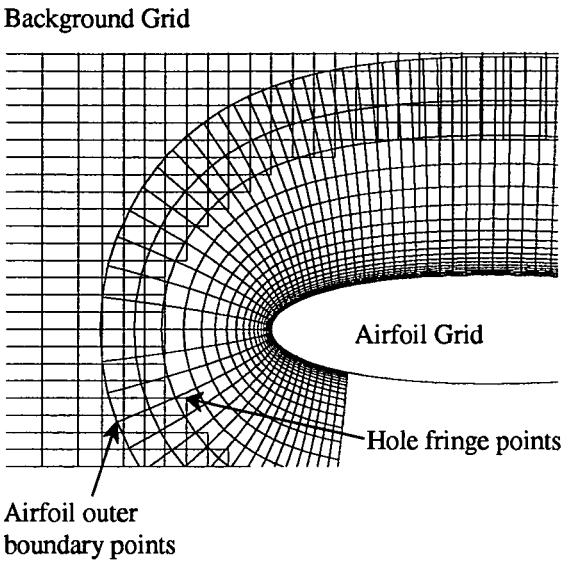


Figure 1. An example of intergrid boundary points.

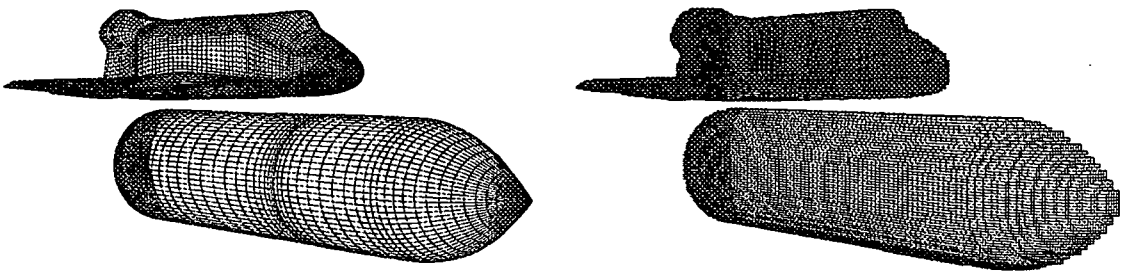
the off-body Cartesian grid, thereby providing the needed off-body to near-body connection for solution information transfer. It is also clear that a similar transfer of information from the near-body solution back to the off-body solution is required. However, the off-body Cartesian grid has no natural boundaries (physical or numerical) that overlap the near-body grid. The Chimera style of overset gridding makes it possible to create an artificial boundary (hole boundary) within the off-body grid system, and thereby establish the required near-body to off-body connectivity. A hole boundary for this case is created by excluding the region of the off-body Cartesian grid that is overlapped by a portion of the airfoil grid (including all of the airfoil). The resulting hole region is excluded from the remaining off-body solution. Conditions for the hole boundary are interpolated from the solution in the near-body airfoil grid. In general, one-way communication connections can be established between any set of component grids through hole and outer boundaries.

IGBP Identification

The task of identifying outer boundary points of overset grid components is trivial and is not discussed further in this paper. However, the task of creating holes in geometrically complex grid systems can be computationally intensive and is not a trivial task. The hole-cutting algorithm described in this section is intended as a replacement for the analytic shape method of hole cutting employed in the DCF3D code[2]. The DCF3D based analytic hole cutters are by far the most efficient method of hole-cutting and IGBP identification of all existing domain connectivity codes[2,3,4,5,6]. However, use of the analytic shape hole-cutters is difficult and requires high levels of user expertise and interaction, thereby precluding automation. The present method is nearly as efficient as the analytic shape approach, and provides the basis for a fully automated system of hole creation and IGBP identification.

The present approach is referred to hereafter as "hole-map" technology. Hole-map technology is based on an idea of the late Professor Joseph Steger that takes advantage of the same search-by-truncation incentives that exists in the inverse-maps employed in DCF3D. Given a system of overset grids, and knowing something of the topology and flow boundary conditions, it is possible to generate approximate hole surfaces associated with each component grid. For example, Figure 2(a) illustrates the no-slip surfaces of the space shuttle external tank and orbiter grid systems. Figure 2(b) illustrates an approximation of the same surfaces defined with respect to a uniform Cartesian system of points. The approximate surfaces shown in Figure 2(b) can be used to carry out inside/outside tests for the determination of IGBPs (intergrid boundary points) far more efficiently than tests associated with the actual hole surfaces. Given the X,Y,Z coordinate of a point in the orbiter, for example, the position of that point within the external tank hole-map can be identified by simple truncation. Once this is known, the field or hole status of the X,Y,Z point in question is determined by the corresponding status of the hole-map element that bounds it.

Hole-maps need to be created only once, even for moving body problems, and can be done automatically given the set of overset component grids and a set of grid index ranges which define hole-cutting surfaces. Default hole-cutter surfaces are



(a) True hole surface. (b) Approximate hole surface.

Figure 2. Comparison of true hole surface and approximate hole surface.

the grid surfaces associated with flow solver no-slip (flow tangency for Euler applications) boundary conditions. Whether the hole-cutter surfaces are supplied by default, or by user specified sets of grid indices, the method of hole-map creation is otherwise fully automatic. The method of hole-map creation developed here consists of four steps.

Step-1. Create a Cartesian box that bounds the given hole surfaces. The size of the box is determined by the bounding minimum and maximum values of hole surface coordinates in the inertial frame X, Y and Z directions. The resolution of a given hole-map is determined as a function of the resolution associated with the given hole surfaces. Again, hole surfaces can be the default surfaces (flow-solver no-slip surfaces), user-specified ranges of grid indices, or user-specified offset distances from the default surfaces. It is important to note that the memory requirements for hole-maps are minimal. The X,Y,Z coordinates for the uniform Cartesian grid associated with the hole-map do not need to be stored. Only the coordinates of the box diagonal points and the spacing associated with the map need to be stored. The hole-map itself is an integer array consisting of values of 0 and 1 depending on whether individual hole-map elements correspond to a point inside, or outside the hole to which the map is associated.

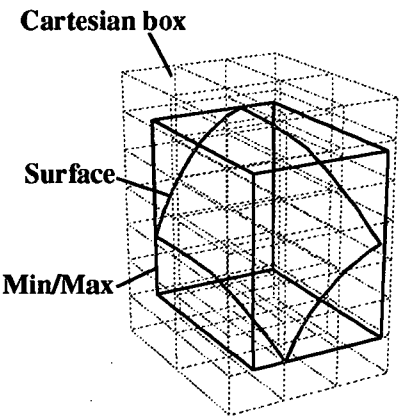
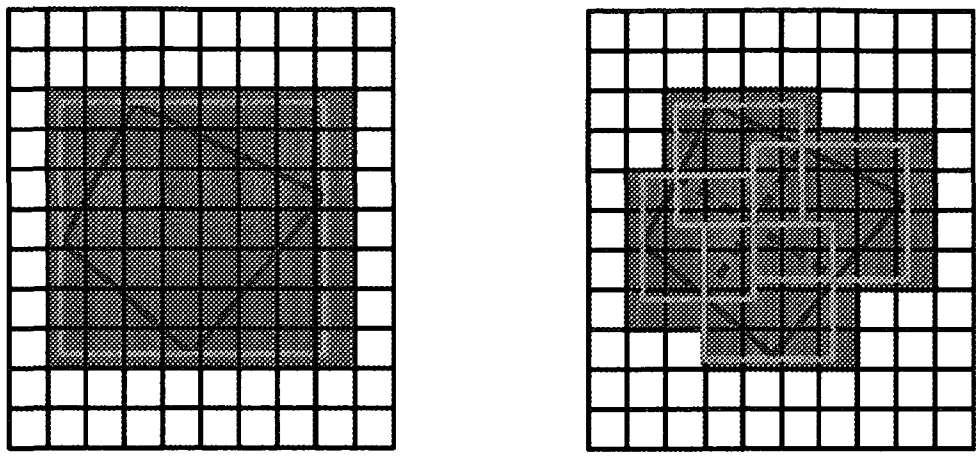


Figure 3. Method of finding hole boundary in a Cartesian box.

Step-2. Find hole boundary within the uniform Cartesian grid associated with the hole-map. This is done by bounding each quadrilateral of the specified hole surface with a min/max box. Then, all points in the hole-map are marked with a 0 or 1 to indicate whether, or not, they are part of the hole boundary (see Figure 3). To maintain the fidelity to the original geometry, each quadrilateral surface element is usually subdivided to avoid marking too many hole-map elements as hole boundary points. Figures 4(a) and 4(b) illustrate this point by subdividing a specified surface element, the number of extra hole-map elements marked as hole boundary points is reduced.

Step-3. Close the surfaces bounded by open boundary curves. It is important for specified hole-surfaces to represent closed, or "water-tight" surfaces. However, in general, the default hole-cutter surfaces for a set of overset component grids, or even user-specified hole-cutter surfaces will not always result in closed composite surfaces. For example, a problem which involves a plane of symmetry would preclude definition of a closed surface around a body centered at the symmetry plane. Accordingly, in the present approach, hole-cutter surfaces are closed according to the following



(a) Original surface patch. (b) Sub-division of surface patch.

Figure 4. Comparison of single surface patch and multiple surface patches approximation.

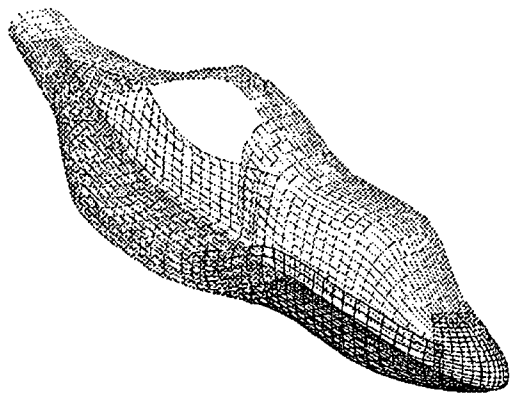
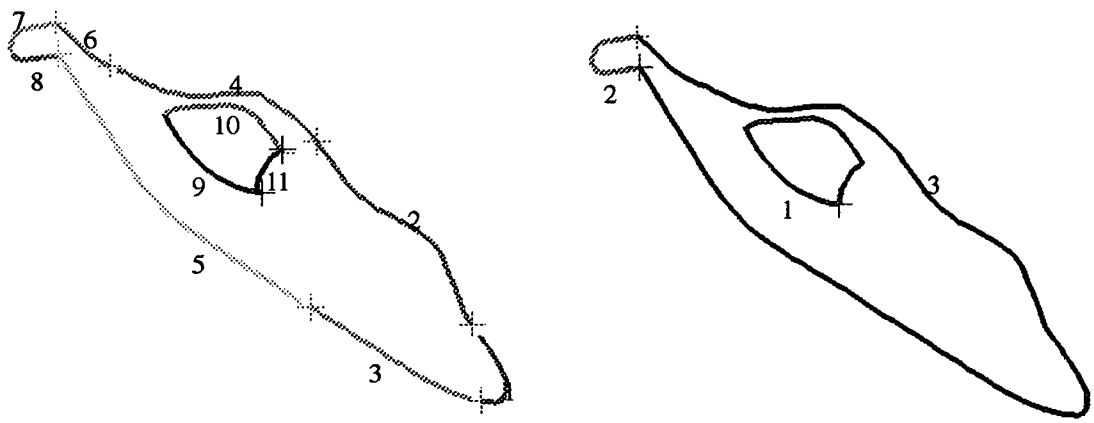


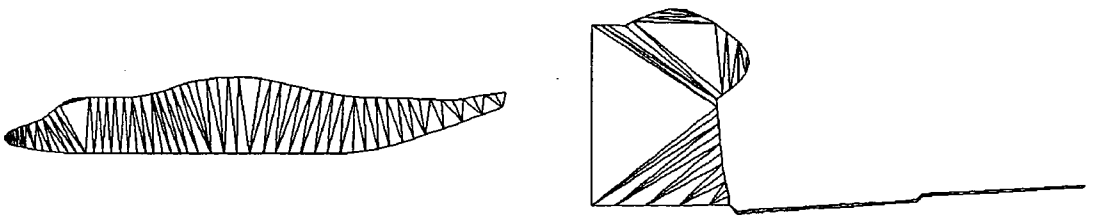
Figure 5. V-22 tiltrotor fuselage surface grid.



(a) Initial open boundary curves (11 segments) (b) Final open boundary curves (3 segments)

Figure 6. Initial and final open boundary curves.

protocol. First, curves on a specified hole-cutter surface that are associated with open boundaries are identified[7]. Figure 5 shows a set of no-slip surfaces on a tiltrotor fuselage[8]. Figure 6(a) indicates the curves from the surface grid components shown in Figure 5 that are associated with open boundaries. Second, the open boundary curves are ordered to form one, or more, continuous curves. This is necessary since grid components may be given in arbitrary order. Third, open boundary curves are grouped into segments for triangulation of the open surfaces. Each curve segment is projected into a 2-D plane and checked for intersection before performing the triangulation. An alternative projection plane is chosen if intersection is found. Delaunay triangulation was implemented to close the surfaces bounded by open boundary curves. Figure 6(b) shows that the eleven open boundary curves identified in Figure 6(a) have been grouped into three continuous segments for triangulation. Figures 7(a) and 7(b) show the triangulation for the symmetry plane of the fuselage shown in Figure 5 and the backend of the space shuttle orbiter shown in Figure 2(a). The Delaunay triangulation did very well in maintaining the thin trailing edge for the orbiter. Finally, the hole surfaces are used to mark points in the hole-map that are occupied by the triangles resulting from Delaunay triangulation as hole boundaries.



(a) Triangulation for V-22 tiltrotor symmetry plane (b) Triangulation for orbiter backend

Figure 7. Triangulations for surface bounded by open boundary curves.

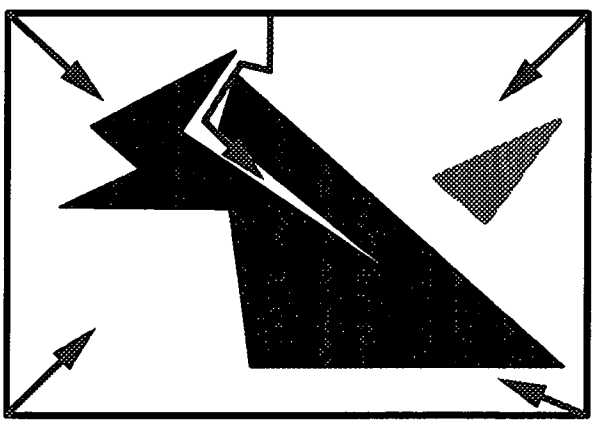


Figure 8. Algorithm to find points inside hole boundaries.

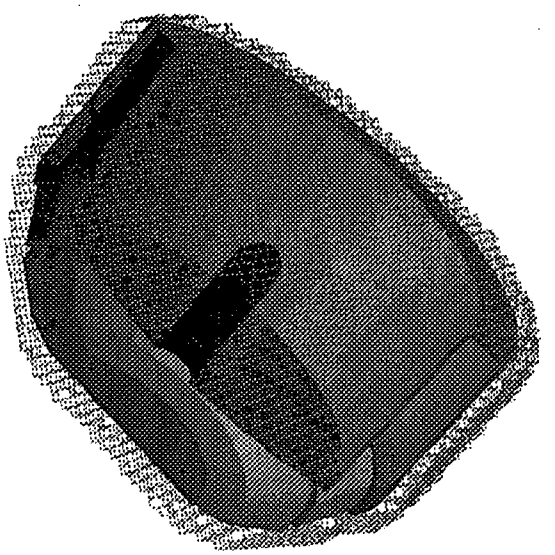


Figure 9. Cartesian approximation for the SOFIA telescope grid.

Step 4. Complete the hole-map definition. Steps 2 and 3 above, marked the hole *surface* in the hole-map. The final step in hole-map creation is to "fill-up" the hole, or more precisely, to identify the *volume* of the hole-map that corresponds to the hole-cutter. The points inside the hole boundary are identified by excluding all the points outside the hole boundary. For simplicity, a 2-D representation of this algorithm is illustrated in Figure 8, though this algorithm is a 3-D algorithm. By closing in from the boundary where the status (inside or outside the hole boundary) of each point is known, one can safely say that a point is outside the hole boundary if any adjacent point is outside the hole boundary. Note that for this algorithm to work, the hole boundary needs to be a closed boundary. The advantage of this algorithm

is multiple.

This hole-filling algorithm is simple and robust. Unlike the common 2-D polygon filling algorithm[9], there is no need to find and sort intersections of the scan line with all edges of the polygon. Better, it eliminates the need to check for vertex intersections if the number of intersections on the sorted list is odd. The approach is applicable for even the most complicated of shapes. For example, Figure 9 shows the Cartesian approximation for the SOFIA telescope grid [10]. The current algorithm easily captures the shape of the original geometry.

The algorithm is easier to vectorize. Since the algorithm does not rely on intersections of scan line to fill the 3-D volume, the vector length is usually longer than the number of points between each pair of intersections.

Given a set of hole-maps for a particular application, creation of Chimera-style holes is trivial. A single pass through each component grid is sufficient to cut all holes and identify all resulting IGBPs (intergrid boundary points). Given the inertial frame X,Y,Z coordinates of any grid point, the hole status of that point can be determined by simple truncation using the set of hole-maps previously generated. Specifically, the hole-map indices of a grid point X,Y,Z can be found from the following equation.

$$J = int\left(\frac{X - X_0}{\Delta X}\right)$$

$$K = int\left(\frac{Y - Y_0}{\Delta Y}\right)$$

$$L = int\left(\frac{Z - Z_0}{\Delta Z}\right)$$

where, (X_0,Y_0,Z_0) is the origin of the hole-map and ΔX, ΔY and ΔZ represent the hole-map resolution in the X, Y and Z coordinate directions. Whether the point is inside, or outside the hole is determined by checking the status of the hole-map element J,K,L.

Donor Element Identification

Given a set of IGBPs, a variety of search procedures are available for locating a corresponding number of suitable donor elements. By far the simplest method to locate an element which bounds a point from the set of IGBPs is an exhaustive search. This consists of testing all possible donor elements to see if the IGBP lies inside. Although the required test is simple, the computational expense required to test every element in every component grid for each point in the IGBP list is prohibitive. Other search procedures that are applicable to this problem include "stencil-walk" (i.e., gradient search), and spatial partitioning methods (octrees, polytrees, etc.). The DCF3D code employs inverse-maps (simple spatial partitioning) in conjunction with a gradient search procedure to locate donor elements. An inverse-map is simply a mapping of the computational spaces of the several overset grid systems onto uniform Cartesian systems of points. Therefore, inverse-maps facilitate the same kind of search by truncation procedure available with the hole-maps described in the previous section. Inverse-maps efficiently identify donor elements to within a small neighborhood of grid points. DCF3D finds the exact donor location via stencil-walking through the small neighborhood. The method of donor identification used in DCF3D has been employed here also.

The objective of the donor identification work here is on optimization or, more precisely, automation via iterative improvement. The inverse-map/stencil-walk approach used in DCF3D is simply the computing engine of the present iterative procedure. Figure 10 shows a schematic of the overall domain connectivity algorithm being suggested here.

There are two motivations for developing an iterative approach to domain connectivity. The first deals with automation of the entire process, significantly reducing the human resources currently needed for completing this task. The second motivation is based on grounds of maintaining solution accuracy. In the latter case, it can be argued that quality rules should involve the flow solution itself. However, the geometric rules of goodness are more strict and, therefore, are sufficient to insure the maximum realizable accuracy from a given set of overset grids.

A necessary input for any method of successive improvement is an initial solution. In the present approach, the initial solution will be derived automatically based on the default hole-cutting surfaces, or those provided by the user. If the default hole-cutting surfaces are used, the initial connectivity solution will correspond to the maximum realizable overlap case for the given set of grids. If valid donor elements (i.e., hole-free donors) cannot be found for all points in the IGBP list, then the given set of grids are flawed, and should be regenerated. IGBPs for which valid donors do not exist indicate regions where grid resolution or overlap is insufficient. Assuming the given set of overset grids are valid for at least the

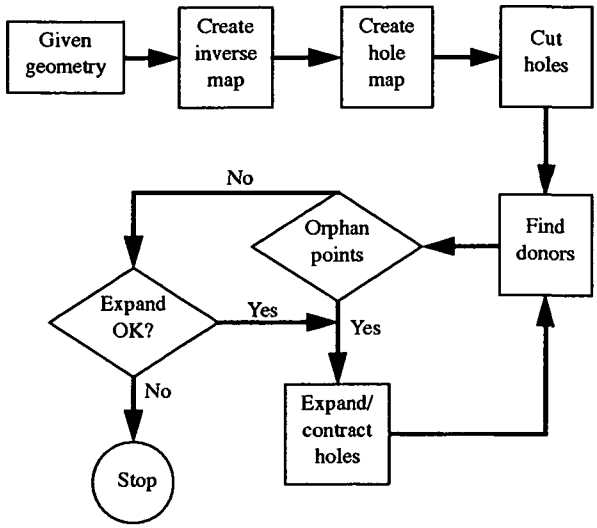


Figure 10. Flow chart of domain connectivity algorithm

default case, the initial solution will be improved according to the iteration indicated in Figure 10. The iteration will continue until the minimum overlap connectivity solution has been obtained.

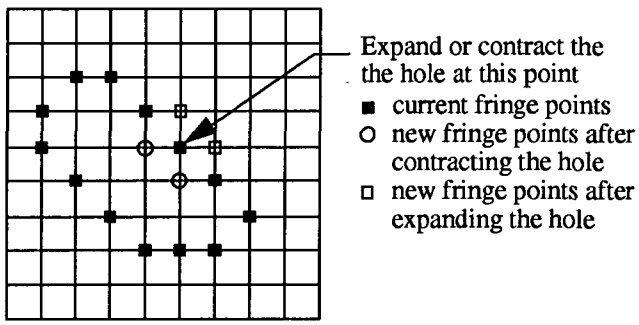
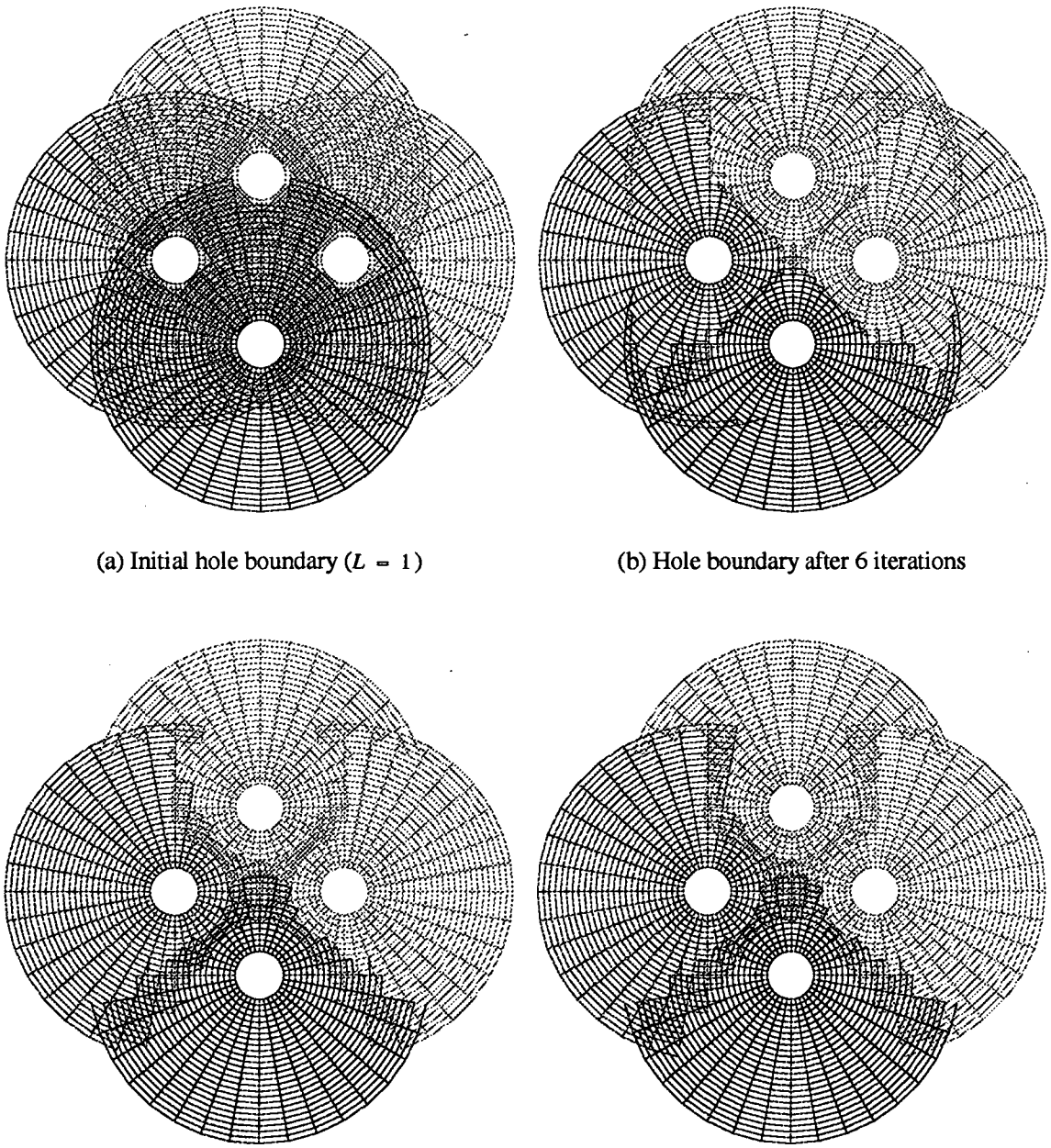


Figure 11. Hole expansion and contraction

The present implementation of the algorithm is simply to expand the hole boundary until an IGBP has been orphaned (i.e., no donor available), or to contract the hole fringe away from the orphaned location until a donor is found. The hole expansion/contraction is performed locally, so at one location, the hole boundary may stop expansion while at other locations, the hole boundary may continue to expand. Figure 11 shows how the hole boundary expands or contracts. The present algorithm satisfies the need for automation of the domain connectivity task. However, since the iteration is currently driven by orphan point detection alone, it does not address the issue of iterative improvement for the purpose of providing the maximum realizable solution accuracy from the given set of component grids. A future implementation of the algorithm will include a more complete set of geometric measures of donor goodness.

The algorithm does not require that the initial connectivity solution be derived from the default hole-cutter surfaces. Rather, any consistent set of hole-cutter surfaces can be used to start the iteration, even if the specified surfaces result in orphaned IGBPs. The hole fringe will be locally expanded/contracted in the vicinity of the orphaned IGBPs. The ability of the algorithm to locally expand and contract in the vicinity of orphaned IGBPs is a very powerful capability. For applications in which there is relative motion between component parts, the initial connectivity solution may be orphan free. However, body motion alters the location of holes and, hence necessitates a new connectivity solution. The present method not only provides the new solution by expanding/contracting hole fringes in advance/behind the moving body, but provides the capacity to control donor quality dynamically throughout the simulation.

Figures 12 and 13 illustrate the flexibility of the algorithm used in dynamic hole expansion and contraction. Though, in



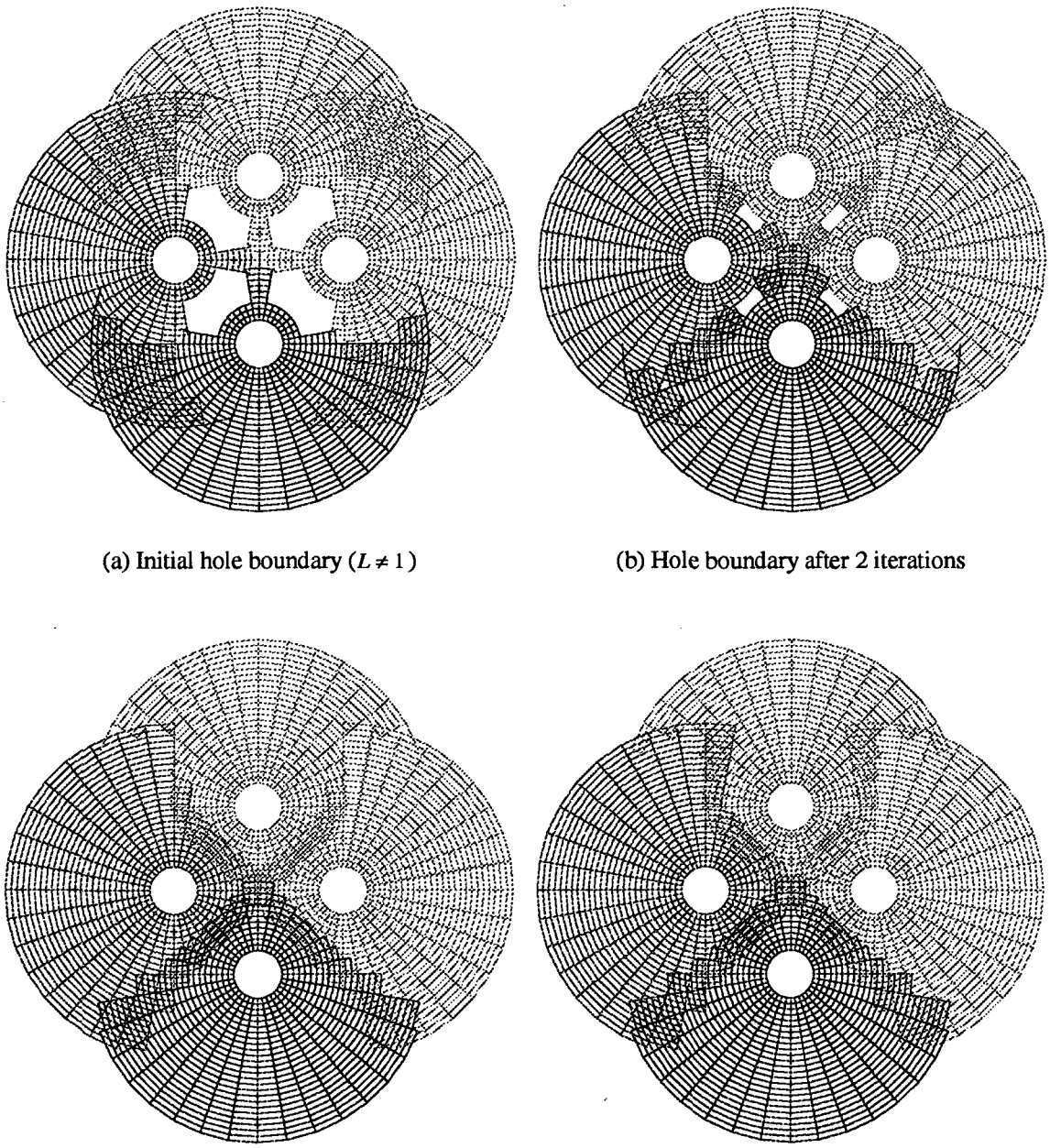
(a) Initial hole boundary ($L = 1$)

(b) Hole boundary after 6 iterations

(c) Hole boundary after 10 iterations

(d) Final hole boundary (14 iterations)

Figure 12. An example of dynamic hole expansion/shrinking starting from no-slip surface.



(a) Initial hole boundary ($L \neq 1$)

(b) Hole boundary after 2 iterations

(c) Hole boundary after 4 iterations

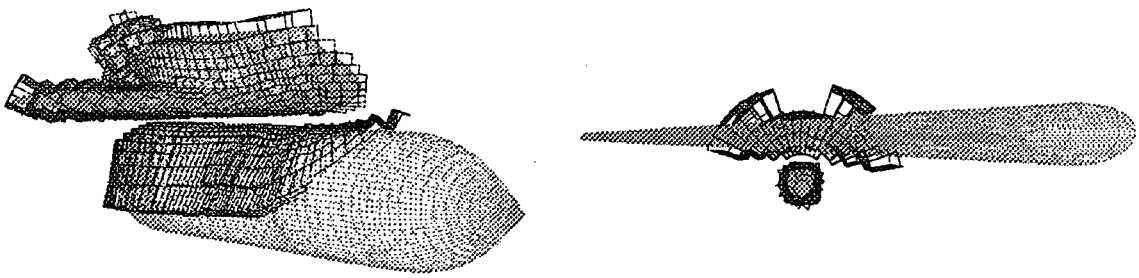
(d) Final hole boundary (16 iterations)

Figure 13. An example of dynamic hole expansion/shrinking starting from a specified hole boundary.

Figure 13, the initially specified hole boundary resulted in empty space in the grid system (see Figure 13(a)), through hole contraction and expansion, the final grid system as seen in Figure 13(d) fills the empty space and minimizes the overlapped area. The final grid systems in Figures 12(d) and 13(d) show a high degree of similarity even though they started with different initial conditions.

RESULTS

Application of the present algorithm using hole-maps and iterative donor identification has been carried out for a set of test cases. The tests were carried out on a range of workstations, including HP 9000/755, IBM RS6000, SGI 4D-210 with 25 MHz MIPS R3000 CPU, SGI Elan with 150 MHz MIPS R4400 CPU, and supercomputers, Cray C90. Figures 14(a) and 14(b) show hole boundaries cut in the shuttle external tank and orbiter grids, and a wing and store grid combination using hole-map technology. Table 1 shows the CPU time usage on SGI 4D-210 with 25 MHz MIPS R3000 CPU for these two cases. The times include all aspects of the domain connectivity process: hole-cutting, donor identification, etc.



(a) ET/ORB hole boundary. (b) Wing/Store hole boundary.

Figure 14. Hole boundaries.

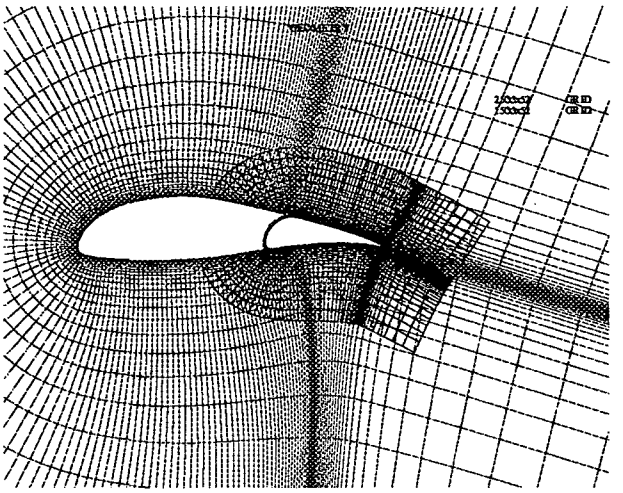
The same wing/store case was also carried out using DCF3D with analytic shape cutters. The CPU time usage is 85.0 seconds on SGI 4D-210 with 25 MHz MIPS R3000 CPU. Thus, the performance of hole cutting using a hole-map is roughly the same as that realizable using analytic shape cutters.

	# of Points	IGBPs	CPU Time on SGI 4D-210 w/ R3000
ET/ORB	588,240	7,628	155.4
Wing/Store	256,863	3,114	79.4

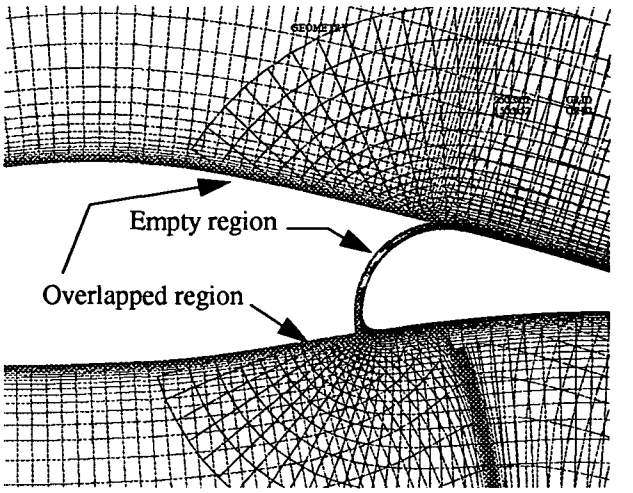
Table 1. CPU time usage for ET/ORB and Wing/Store testcases.

Figure 15(a)-(d) illustrate the domain connectivity capability of DCF3D in regions where component grids are close to each other. This testcase was carried out with dynamic hole expansion/contraction. As can be seen in the close-up view of initial hole cut, there are empty regions between the wing and the flap. These empty regions were filled through hole expansion in the final hole cut. Also, much of the overlapped area was removed in the final hole cut.

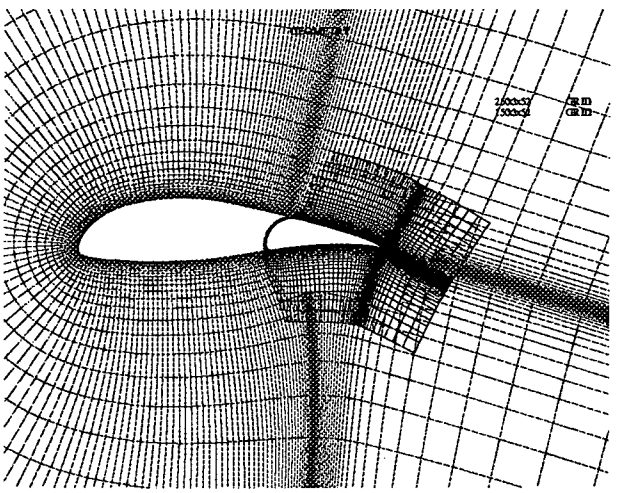
A generic 747 wing/fuselage configuration was carried out as a benchmark test to compare the performance of DCF3D and Pegasus. Figure 16 shows the grid connection between wing root and fuselage computed by DCF3D. Part of the fuselage surface was cut out by the wing root grid. The CPU time usage on Cray C90 for DCF3D and Pegasus is 188.80 and 436.24 seconds respectively. Both DCF3D and Pegasus were run twice because the hole fringe points on the fuselage surface had to be projected to wing root grid after the fuselage surface was cut by wing root grid in the first run. In the second run for DCF3D, inverse maps were recreated, which may not be necessary since the grid points on the fuselage surface were moved a tiny distance after projection. Without creating the inverse map for the second run, the CPU time would have been reduced by 30 seconds. The CPU time could have been reduced further if DCF3D restart option was used to take advantage of known solutions obtained in the first run. Figures 17(a)-(j) show the comparison of wing cross section pressure coefficients at various wing spanwise locations for the flow solutions using composite grids generated byDCF3D



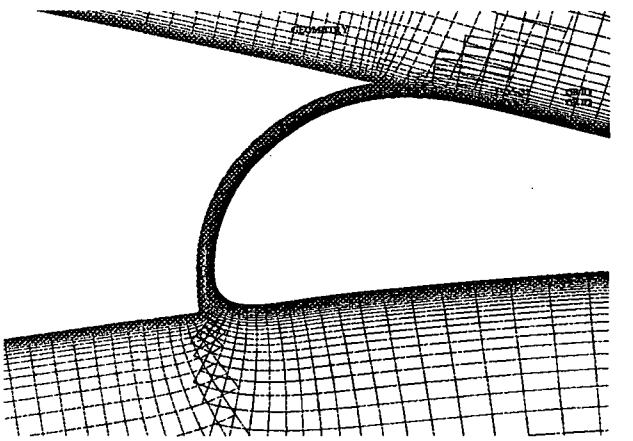
(a) Initial hole cut.



(b) Close-up view of initial hole cut.



(c) Final hole cut at iteration 15.



(d) Close-up view of final hole cut.

Figure 15. Hole boundaries for multi-element airfoil.

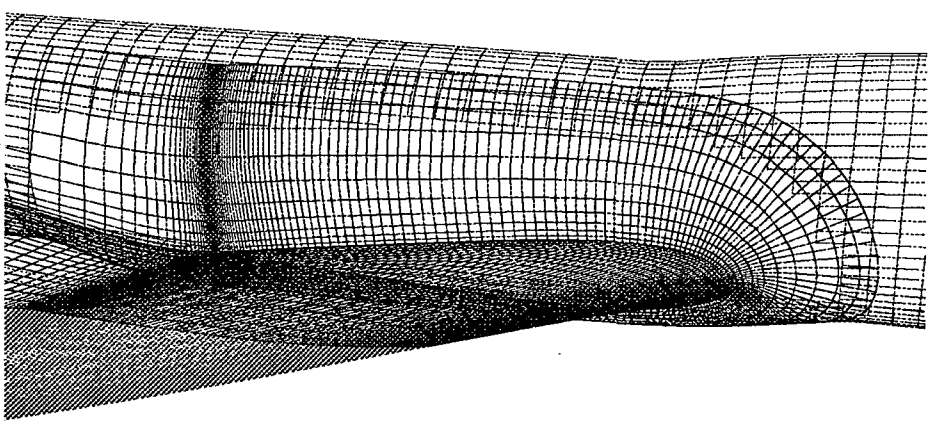
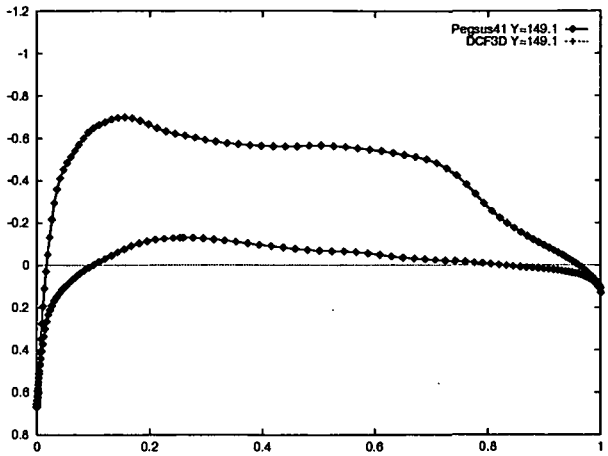
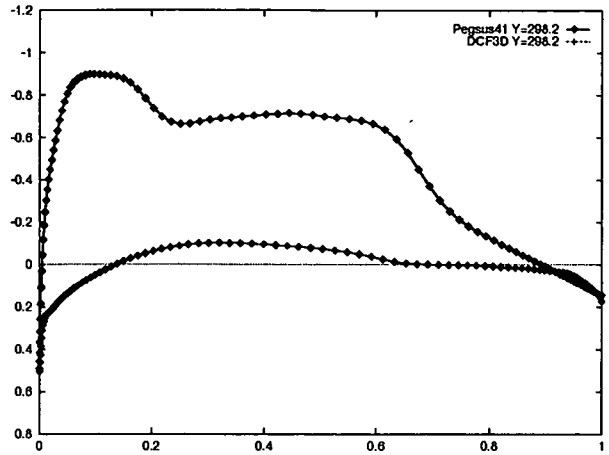


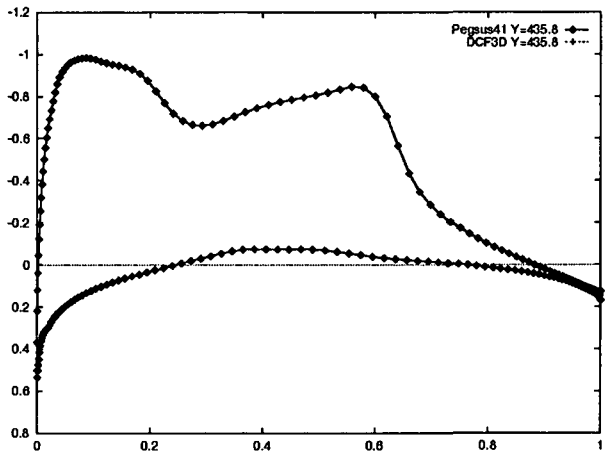
Figure 16. Grid connection at the wing root and fuselage junction for generic 747 configuration.



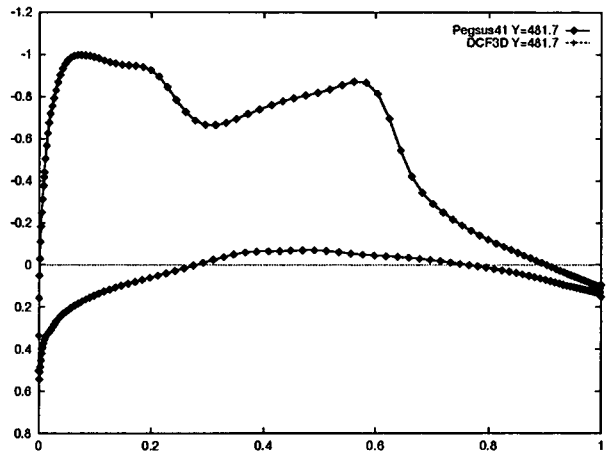
(a) 2.1% wing span



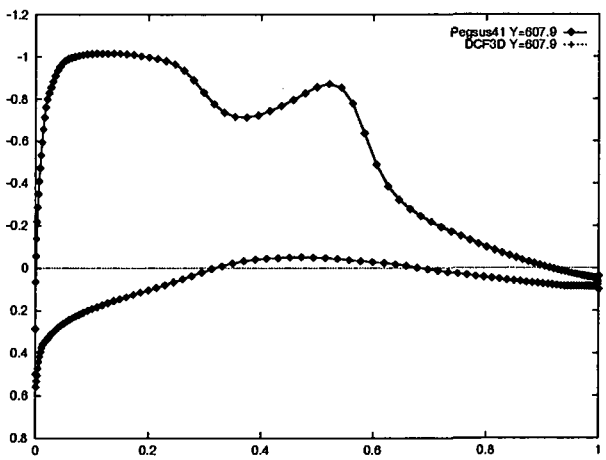
(b) 16.5% wing span



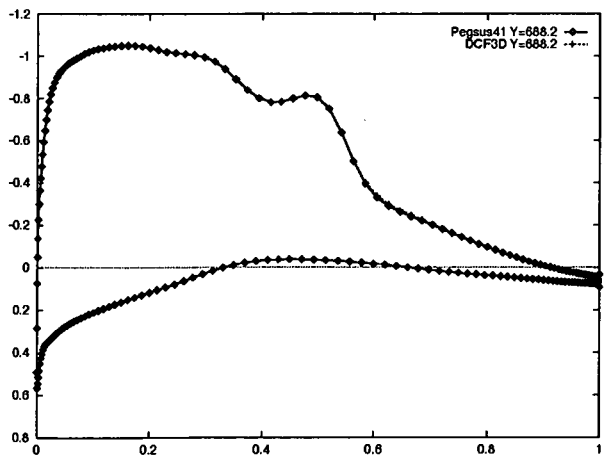
(c) 29.8% wing span



(d) 34.3% wing span

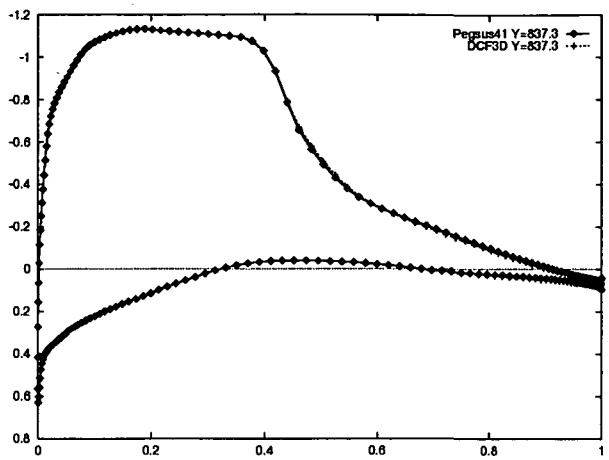


(e) 46.5% wing span

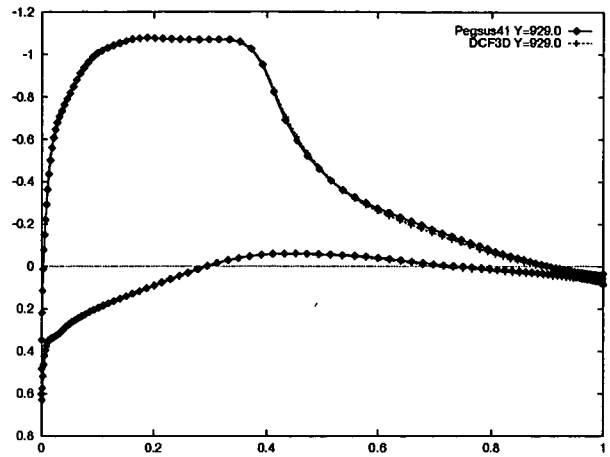


(f) 54.2% wing span

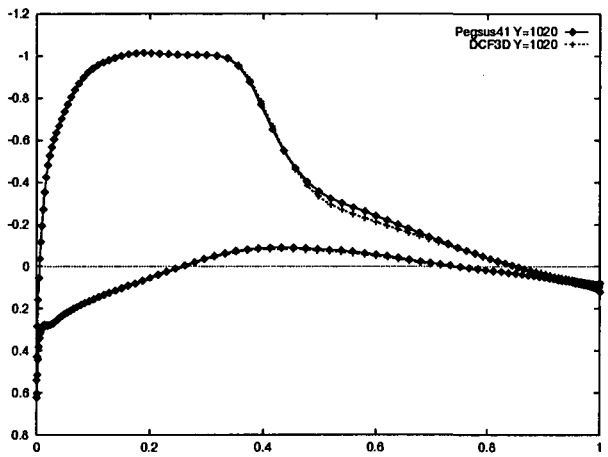
Figure 17. Cp comparison for generic 747 wing/fuselage configuration.



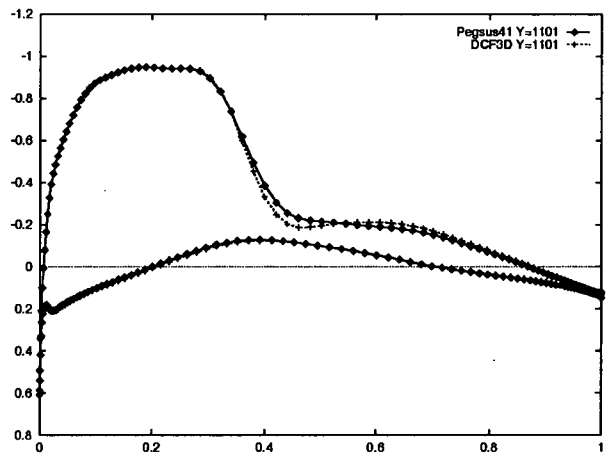
(g) 68.7% wing span



(h) 77.5% wing span



(i) 86.3% wing span



(j) 94.2% wing span

Figure 17. (continued)

and Pegasus. The pressure coefficients show no visible difference up to 68.7% wing span. At 86.3% and 94.2% wing span DCF3D composite grids gave a stronger shock than the corresponding Pegasus grids. This may be attributed to the fact that the intergrid boundary near the wing tip of DCF3D composite grids is closer to the wing surface than that of Pegasus composite grids'.

CONCLUSIONS

Novel algorithms for creating Chimera-style hole boundaries and iterative improvement of domain connectivity solutions have been presented. The algorithms have been incorporated into the DCF3D code and tested on a set of applications. The hole-map method of creating Chimera-style hole boundaries was shown to be as efficient as a very efficient method based on analytic shapes, and shown to represent a key component for a fully automated algorithm for domain connectivity. The algorithm for iterative improvement of domain connectivity solutions for systems of overset grids was demonstrated. The potential of the method for insuring the maximum realizable accuracy from a given set of overset grids was shown. This potential exists even for applications which involve relative motion between component parts. In the present implementation of the algorithm, only rudimental rules for donor goodness were employed. A more complete set of rules for measuring donor quality will be developed and implemented in the future.

ACKNOWLEDGEMENTS

This work was supported under NASA ARC grant NCC2-783. All computational results reported in this paper were carried out on the NAS facility at NASA ARC. The authors wish to acknowledge the influence of the late Professor Joseph Steger on this work. In addition, thanks are due to Drs. William Chan and Cathy Maksymiuk for valuable discussions on the issues presented herein. The support and direction provided by Dr. Pieter Buning is also gratefully acknowledged.

REFERENCES

[1] Steger, J., Dougherty, F. C., and Benek, J., "A Chimera Grid Scheme," Advances in Grid Generation, K. N. Ghia and U. Ghia, eds., ASME FED-Vol 5., June 1983.

[2] Meakin, R., "A New Method For Establishing Inter-Grid Communication Among Systems of Overset Grids," AIAA Paper 91-1586, June, 1991.

[3] Benek, J., Donegan, T., and Suhs, N., "Extended Chimera Grid Embedding Scheme with Application to Viscous Flows," AIAA Paper 87-1126-CP, 1987.

[4] Dietz, W., Kacocks, J., Fox, J., "Application of Domain Decomposition to the Analysis of Complex Aerodynamic Configurations," SIAM Conf. Domain Decomposition Meths., Houston, TX, March 1989.

[5] Brown, D., Chesshire, G., Henshaw, W., and Kreiss, O., "On Composite Overlapping Grids", 7th Internat. Conf. on Finite Element Methods in Flow Problems, Huntsville, AL, April 1989.

[6] Maple, R.C. and Belk, D.M., "Automated Set Up of Blocked, Patched, and Embedded Grids in the Beggar Flow Solver," Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Ed. N.P. Weatherill et al., 1994, Pine Ridge Press, pp. 305-314.

[7] Chan, W.M. and Buning, P.G., "A Hyperbolic Surface Grid Generation Scheme and Its Applications," AIAA Paper 94-2208, June, 1994.

[8] Meakin, R., "Moving Body Overset Grid Methods for Complete Aircraft Tiltrotor Simulations," AIAA Paper 93-3350, July 1993.

[9] Foley, J.D. and Van Dam, A., "Fundamentals of Interactive Computer Graphics," Addison-Wesley, Reading Mass., 1983.

[10] Atwood, C.A. and Van Dalsem, W.R., "Flowfield Simulation about the SOFIA Airborne Observatory," AIAA Paper 92-0656, 1992