

PEGASUS 5: AN AUTOMATED PRE-PROCESSOR FOR OVERSET-GRID CFD

Norman E. Suhs

U.S. Army

Aviation & Missile Research, Development and Engineering Center

Redstone Arsenal, AL

Stuart E. Rogers

NASA Ames Research Center

Moffett Field, CA

William E. Dietz

Flow Analysis, Inc.

Tullahoma, TN

Abstract

An all new, automated version of the PEGASUS software has been developed and tested. PEGASUS provides the hole-cutting and connectivity information between overlapping grids, and is used as the final part of the grid generation process for overset-grid computational fluid dynamics approaches. The new PEGASUS code (Version 5) has many new features: automated hole cutting; a projection scheme for fixing gaps in overset surfaces; more efficient interpolation search methods using an alternating digital

tree; hole-size optimization based on adding additional layers of fringe points; and an automatic restart capability. The new code has also been parallelized using the Message-Passing Interface standard. The parallelization performance provides efficient speed-up of the execution time by an order of magnitude, and up to a factor of 30 for very large problems. The results of three example cases are presented: a three-element high-lift airfoil, a generic business jet configuration, and a complete Boeing 777-200 aircraft in a high-lift landing configuration. Comparisons of the computed flow fields for the airfoil and 777 test cases between the old and new versions of the PEGASUS codes show excellent agreement with each other and with experimental results.

Introduction

The overset grid approach, also called the chimera grid-embedding scheme, has been developed and refined for over 15 years. The strategy of this approach is to break a complex computational domain into smaller regions that can each be represented by relatively simple grids. With this approach comes a major hurdle in creating the data structure that specifies the inter-connectivity among the grids. The initial creators of the chimera grid-embedding scheme (Ref. 1) developed the PEGASUS code to establish the data structure required for communication among the overlapping structured meshes. Depicted in Fig. 1 is the relationship between PEGASUS and the flow solver. Each mesh that makes up the domain is input into PEGASUS to determine the inter-connectivity among the meshes, i.e., the interpolation data. The interpolation data that is passed to the flow solver includes a list of the mesh points that are interpolated, the associated interpolation coefficients, and the donor cell for each interpolated point. Also included in the interpolation data is a list of the points that are removed (i.e., blanked) from the computational domain due to the fact that they are interior to a solid body. These points are also known as hole points. The user inputs depicted in Fig. 1 specify how this interpolation data is created.

There are several codes that perform the same basic functions as PEGASUS. These include DCF3D (Ref. 2), Beggar (Ref. 3), FASTRAN (Ref. 4) and Overture (Ref. 5). Each of these codes produces some type of interpolation data that is used by a flow solver. Additionally, each code has some level of automation to ease the

creation of the interpolation data. PEGASUS has been successfully used with OVERFLOW (Ref. 6), NXAIR (Ref. 7), INS3D (Ref. 8) and a cell-centered code, DXEAGLE (Ref. 9).

As Computational Fluid Dynamics (CFD) has matured and aerodynamic configurations of interest have increased in complexity, the grid systems used in overset methods have also become more complex and have required greater numbers of grids for accurate representation. With earlier versions of PEGASUS, the user input required for accurate hole cutting and interpolation increased dramatically with the geometric complexity of the configuration. This placed a high premium on automation, which defined the major impetus for the development of the latest version of PEGASUS. This paper presents a background of the process PEGASUS performs to produce the interpolation data, along with a detailed discussion and demonstration of the automation of this process.

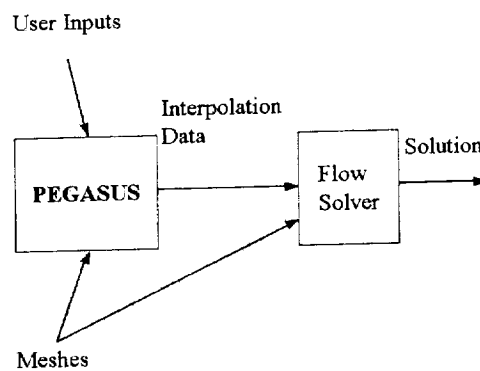


Fig. 1. PEGASUS/Flow Solver Relationship.

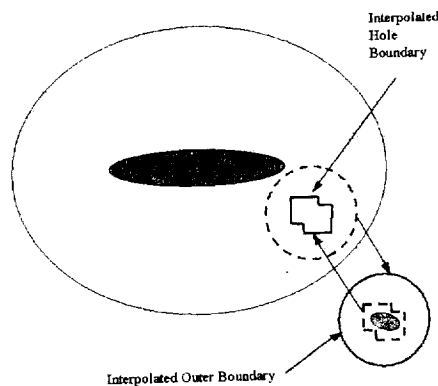


Fig. 2. General Concept of Overlapping Meshes

Background

The general concept behind the chimera approach is illustrated in Fig. 2, which depicts two independently generated meshes representing a flapped airfoil. The flap mesh is embedded within the airfoil mesh. Clearly, the flap mesh outer boundary can receive flow-field information interpolated from appropriate mesh cells (often referred to as interpolation stencils) within the airfoil mesh. However, a reverse process must occur as well; the airfoil mesh must receive flow-field information from the flap mesh. An artificial boundary must be defined within the airfoil mesh, since certain points within the airfoil mesh are interior to the flap and thus do not lie within the airfoil/flap flow field. The artificial boundary points of the airfoil mesh that are fully contained within the computational region of the flap mesh can be updated by interpolation from mesh cells of the flap mesh. Generally, any mesh can receive information from other meshes through outer boundary and artificial boundary points.

The interpolation process is further illustrated in Fig. 3, which depicts a portion of the overlap region between the airfoil and flap meshes. Airfoil mesh points that are contained within a certain region surrounding the flap are excluded from the computational domain of the airfoil mesh. (In chimera terminology, they are "blanked" points, or hole points.) The exclusion of points is accomplished by defining a hole creation boundary within the flap mesh that will define the region within which all airfoil points are to be blanked. The points in the airfoil mesh surrounding the blanked points are hole fringe points; they receive flow-field information interpolated from mesh cells within the flap mesh. Correspondingly, points on the outer boundary of the flap mesh receive flow-field information interpolated from mesh cells within the airfoil mesh.

[NOTE: Fig. 3 is missing the label for the flap-grid outer boundary, this will be fixed.....]

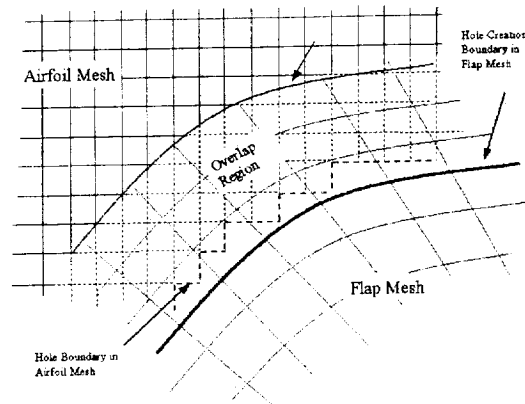


Fig. 3. Detail View of Overlap Region

History of PEGASUS

The PEGASUS code has been a main component of the overset grid methodology since its inception and has gone through many upgrades, increasing its generalization, speed, flexibility, and automation. The first version of PEGASUS (Ref.1), had limited connectivity and hole cutting capabilities. In particular, a grid hierarchy was imposed, i.e., a grid could only cut a hole in a larger grid and could only interconnect with this larger grid. Simple overlapping, with hole cutting, was not allowed. Additionally, the hole cutter was limited to specific types of topologies. If a new topology was required, the PEGASUS code had to be modified to accept this new topology.

Version 2 of PEGASUS (Ref. 10) added more generalization to the interconnectivity. With this generalization there were no restrictions on the interconnectivity among the grids. Grids were allowed to cut holes in any number of grids and to overlap or overset any number of grids. Still, hole cutters were limited to specific topologies.

In version 3 of PEGASUS (Ref. 11), the hole cutting methods were generalized. Greater control was given to the user in creating holes. Holes could be created by any number of surfaces from a single grid. This greatly increased the complexity of geometries that could be handled. Additionally, the NAMELIST user input was improved.

Version 4 of PEGASUS (Ref. 12) continued the improvements in hole cutting by allowing surfaces from multiple grids to create a hole. Also, PEGASUS 4 included a restart capability that was particularly useful for moving body problems.

Motivation for PEGASUS Version 5

As problem sizes increased, the number of inputs required in the version 4 PEGASUS input file increased significantly. For example, a particular high-lift aircraft case (Ref. 13) that contained 153 meshes and 33 million grid points required over 17,000 lines of PEGASUS input. Generating this input file required many weeks of work by an experienced user; this much input also created the potential for many user input errors. In 1996 the NASA Advanced Subsonic Technology/Integrated Wing Design program set a goal of reducing the amount of time to produce a solution for an entire aircraft in a high-lift configuration. The automation of PEGASUS was one of the objectives of this project, such that the user would have to provide little or no input to the code. This paper is a presentation of the results of this effort by the authors. Unlike earlier versions of PEGASUS, which borrowed heavily from earlier versions, PEGASUS 5 implements a completely redesigned approach. An entirely new code was written from scratch, implementing new features and algorithms designed to automate the process of oversetting structured overset grids. The new version was written in Fortran90, to take advantage of dynamic memory allocation and programming features, such as pointers, that lend themselves to particular requirements of establishing domain connectivity.

The following sections describe the code features, including minimization of user input, automatic hole cutting, methods used for searching and selection of interpolation points, optimization of the holes and overlapping grid regions, projection of overlapping viscous grid surfaces, and the automatic restart procedure. A description of the parallelization of the code is presented. Finally, three computational examples are presented, including comparisons of flow-solutions obtained using the grid systems generated by both the old and the new PEGASUS codes.

Automation of the Oversetting Process

There are three primary operations that PEGASUS performs to create the interpolation data required by the flow solver. The first of these steps is hole cutting. The mesh points that are within a solid body must be identified, so that they can be removed from the computational domain by the flow solver. Looking back at Fig. 2, the airfoil mesh points that are contained within the flap mesh must be identified. For two-dimensional grids, this process appears to be relatively easy, but for three dimensions and multiple overlapping meshes the hole cutting process can be difficult.

The second step is to identify the interpolation points. There are two types of interpolation points: hole-fringe points and outer-boundary points (see Figs 2 and 3). The hole-fringe points are easily identified as any point that has a hole-point as a neighbor. An outer-boundary point is any point which lies on the boundary of a computational mesh and which will not be updated by a boundary condition within the flow solver.

The third step is the identification of the donor cells that will be used to update the interpolated fringe and boundary points identified in the previous step. If a suitable donor cell cannot be found for an interpolation point, the point is termed an "orphan".

The first two of these steps requires knowledge of the complete set of boundary conditions that are to be applied by the flow solver onto each mesh. The PEGASUS 5 code, therefore, requires the flow solver boundary conditions as part of its input. The format and definitions of the boundary conditions for this input was adopted from the OVERFLOW code. Some grid-generation software provides a mechanism to produce an OVERFLOW input file automatically. For example, the OVERGRID package (Ref. 14), which is part of the Chimera Grid Tools (Ref. 15), has a feature that will automatically detect the appropriate boundary conditions for each mesh, and write out both an OVERFLOW input file and a PEGASUS 5 input file.

Hole Cutting

An automatic hole-cutting capability is provided with PEGASUS 5, although the manual hole-cutting approaches in previous versions have been retained to provide additional control over the hole-cutting process. With the automatic hole-cutting approach, a good hole structure often can be developed with little or no intervention by the user.

The automatic hole generation process is illustrated using a two-dimensional example (i.e., a 3-element airfoil). Figure 4 illustrates the solid boundaries of the airfoil, as defined in the PEGASUS 5 input file. The definition of the solid boundaries of the configuration must represent an air-tight surface, with no gaps, holes, or leaks. If the automatically generated or original user-generated boundary conditions do not define an air-tight surface, the user can specify additional boundary conditions in the input file to augment the surface and close any gaps.

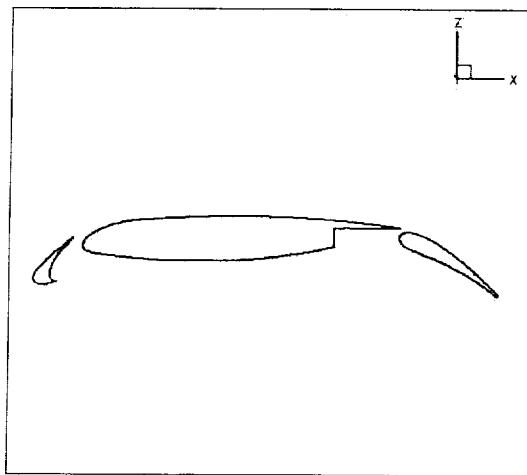


Fig. 4. Three-Element Airfoil Solid Boundaries

The overall objective of the hole-cutting process is to partition the computational domain into "inside" and "outside" regions. In PEGASUS 5, this is accomplished using Cartesian meshes, where it is desired to mark each Cartesian element as an "inside", "outside", or "fringe" element. Spatial partitioning approaches used in previous versions of PEGASUS were based on the use of surface normals. This approach exhibited many situations that had to be dealt with as special cases, particularly when dealing with CFD configurations with surface

discontinuities. Instead, PEGASUS 5 uses a hole-cutting approach that does not depend on surface normal definitions, and therefore can accommodate surface discontinuities.

A Cartesian mesh is generated which fully encompasses the solid boundaries of the configuration. The elements of the Cartesian mesh which intercept the solid surface elements of the airfoil are identified and designated as fringe elements. Some of the fringe elements in the slat-wing region of the 3-element airfoil are depicted in Fig.

5.

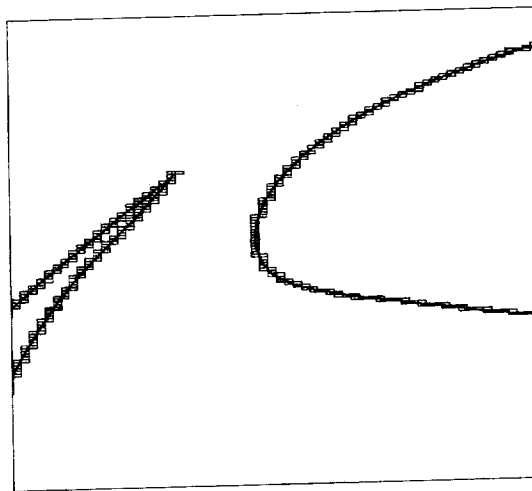


Fig. 5. Fringe Elements

It is assumed that the corner elements of the Cartesian mesh are "outside" elements. Any unidentified (i.e., non-fringe) element that is adjacent to an outside element must itself be an outside element. The outside region is thereby identified by a painting algorithm that marches from the corner elements inward until no more elements that are adjacent to outside elements can be found. The outside region is thereby completely defined, and is depicted in Fig. 6.

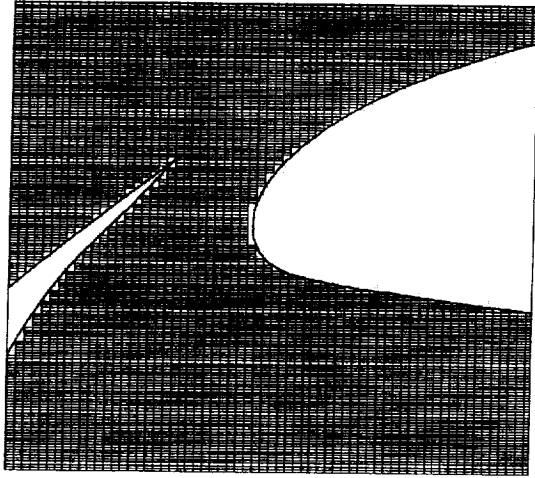


Fig. 6. Outside Elements.

Finally, any element remaining that is not either an outside or fringe element must be an "inside" element. Inside elements are depicted in Fig. 7.

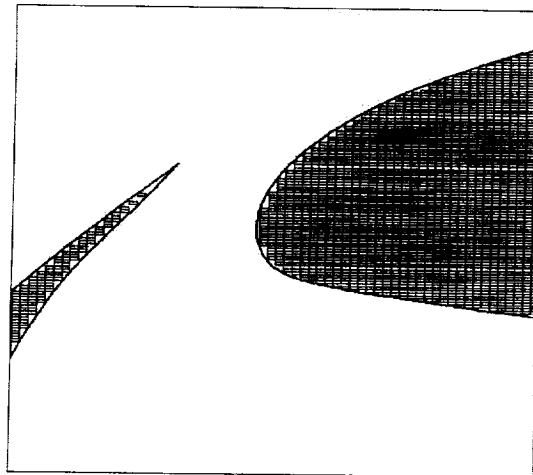


Fig. 7. Inside Elements

The Cartesian mesh is now a completed "hole map". Given an arbitrary grid point, the Cartesian element within the hole map in which the point resides can very quickly be identified. Points that are encompassed by "outside"

or "inside" elements are marked as field points or hole points, respectively. Points that fall within fringe elements can assume either identity, and therefore must undergo further processing. PEGASUS 5 uses a "line-of-sight" algorithm to determine the status of such a grid point. This algorithm tests to see if a clear line-of-sight exists between the point and an outside or inside (i.e., non-fringe) element, and if so, then the point will assume the identity of that element. A clear line-of-sight means that a vector from the point to a neighboring non-fringe element does not intersect the solid surface contained in the fringe element. This algorithm is illustrated in Fig. 8. Points that can "see" an outside element are field points; points that can "see" an inside element are hole points.

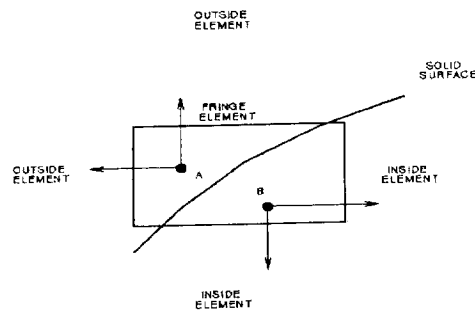


Fig. 8. Line-of-Sight Algorithm

In the example of Fig. 8, Point A is outside. Point B is inside, and will be marked as a hole point.

Outer Boundary Specification

The automation of the outer-boundary point specification is straightforward, since all boundary conditions have been supplied in the input file and are available to PEGASUS 5. The minimum and maximum index surfaces that have not been specified as boundary conditions for the flow solver are designated as the outer boundaries, and these points are added to the list of points that require interpolation stencils.

It can be desirable to have two layers of interpolation points at the hole fringes and at the outer boundaries. The number of layers of interpolation points is known as the "fringe level". A fringe level of two has certain advantages within the flow solver and can produce more accurate solutions. In the new code the user can set the

fringe level for holes and outer boundaries with a single input, or set the fringe level for holes and outer boundaries separately. It should be noted that not all flow solvers accommodate a mixed single/double fringe level of interpolated boundary points.

Boundary Point Interpolation

The identification of hole and outer boundary points is a starting point for the overlap optimization procedure employed by PEGASUS 5. With optimized overlap, many points interior to the grid may ultimately be identified as interpolated boundary points. Therefore, the interpolation process in PEGASUS 5 begins by searching for all possible donor cells from all grids for every single grid point. This entire process is broken down into sub-processes, each involving a pair of grids, one as the donor, the other as the recipient. Note that for any two grids A and B, there are two sub-processes: one with grid A as the donor and grid B as the recipient; the other with grid B as the donor and grid A as the recipient. There are $N*(N-1)$ possible grid pairs, where N is the number of grids in the configuration. Since the number of grid pairs grows as the square of N, the interpolation approach used by PEGASUS 5 places a high premium on the efficiency of the interpolation process.

The interpolation sub-process for a given grid pair begins by testing the intersection of the two meshes, using several different Cartesian and rotated Cartesian boxes. For each grid, these boxes are the smallest box that fully surrounds all of the grid points. If the boxes of the two different grids do not intersect, then no interpolation between the grid pairs is possible. The sub-process next loops through every single grid point in the recipient grid. Inside this loop it first tests to see if the grid point is inside the Cartesian boxes of the donor grid, and discards the point if it is not. It then proceeds by searching for a donor grid cell that is "close" to the final interpolation cell in the donor grid. This is accomplished efficiently through a spatial partitioning scheme; the approach used in PEGASUS 5 is based on a data structure known as an Alternating Digital Tree (ADT), which is described in Ref. 16. ADT structures are generated and stored for each mesh at the beginning of the program's execution. Given a grid ADT and the recipient grid point, a "close" cell in the donor grid can be found very quickly.

Once a close donor cell is identified, a stencil-jumping algorithm is used to find the donor cell which contains the recipient point. The stencil-jumping inverts the equations for tri-linear interpolation using a Newton iteration. This solves for a delta in the computational index space which will point to a donor cell that contains the recipient point. The Newton iteration generally requires a small number of iterations (~3-5), and the stencil-jumping will typically converge in two or three jumps if the initial guess is close and the grid is smooth.

The stencil-jumping procedure has some additional enhancements to improve its robustness. One such enhancement is the ability to detect and to "jump" across the computational boundary in a C-grid or in a periodic O-grid. This is important because the initial starting point returned by the ADT may be close in physical space to the final interpolation element, but may be much farther away in computational space.

Cell Difference Parameter

It is common in an overset grid system to have three or more grids overlapping in the same physical space. Therefore, it is common that a particular boundary or fringe recipient point to have two or more possible donor cells. A new algorithm has been implemented in PEGASUS 5 which is used to determine which of the possible donor cells to select in this case. Previous versions of the PEGASUS code required the user to supply a prioritized link list for each grid. This list specified which grids could act as donor grids for the given recipient grid and all of its points. This approach not only required a lot of detailed input from the user, but it also unnecessarily constrained the choices for the donor interpolation cells. The current approach avoids the global constraint for each mesh, and instead examines the local cells in each individual case.

Experience has shown that the accuracy of a CFD simulation can be degraded when the size of the donor grid cells in the overlapping region are a significantly different size than the recipient cell. This is due to the disparate abilities of a coarse mesh to resolve a flow gradient as compared to a finer mesh. Based on this observation, the new algorithm selects the best donor cell using a measure of the difference in size and orientation between the donor and recipient cells. A qualitative measure of this difference has been developed, and is called the cell-difference parameter (cdp). The cdp is defined as:

$$CDP = \sum_{j=1}^3 \frac{(X_{DB_j} V_B - X_{DI_j} V_I)}{X_{DB_j} V_B}$$

X_{DB_j} = Maximum component of the diagonals of the boundary cell

V_B = the volume of the boundary cell

X_{DI_j} = Maximum component of the diagonals of the interpolated cell

V_I = the volume of the interpolated cell

Values for the cell difference parameter will vary from 0 (the best) to very large values. Examples of the cdp are shown in Fig. 9 for different pairs of two-dimensional cells.

















Interpolated Boundary Point Cell	Interpolation Stencil	Cell Difference Parameter
		0.0
		0.0
		0.5
		1.0
		2.0
		0.5
		9.9
		4.9

Fig. 9. Examples of Cell-Difference Parameter

Overlap Optimization

The final step in creating a good solution to the connectivity of overset grids requires some type of overlap optimization. In previous versions of PEGASUS, it was left up to the user to determine how much overlap to leave between neighboring grids and where the overlap boundaries should occur. This required a significant amount of user expertise and time. PEGASUS 5 provides an automated approach to optimizing the overlap. Again, a new algorithm has been developed on the premise that the donor and the recipient interpolation cells should be of similar size.

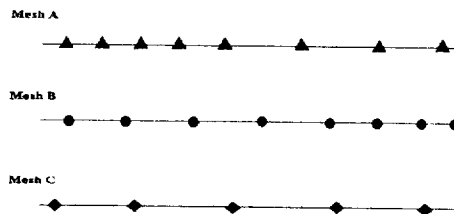
The overlap optimization process in PEGASUS 5 is robust and requires no user interaction. This process is performed after the automatic hole-cutting and the outer boundary points and their donor cells have been identified.

The overlap optimization method is based on a philosophy that the finest mesh points are kept as part of the computational domain while the coarser mesh points should be interpolated from the finer mesh points. To demonstrate the steps required to achieve the overlap optimization, three one-dimensional meshes will be used (see Fig. 10a). Mesh A is stretched from fine to coarse, Mesh B is stretched from coarse to fine, while Mesh C has constant spacing that is coarser than both Mesh A and Mesh B.

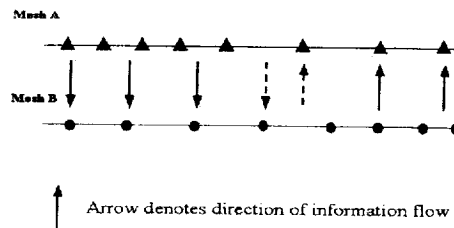
The first step is to interpolate between the mesh pairs. Starting with Meshes A and B, Mesh A interpolates all points from Mesh B and Mesh B interpolates all points from Mesh A. Then, only the coarser interpolated mesh points (i.e., those points that are interpolated from finer mesh regions) are kept (see Fig. 10b). The arrows in this figure and the remainder of Fig. 10 indicate the direction of the data flow. The head of the arrow points to the interpolated point, while the tail indicates the cell that donates data to the interpolated point.

In step 2, the interpolated points identified in step 1 are checked to determine if they are also part of a donor cell. If an interpolated point is part of a donor cell, it is removed as an interpolated point. The result for Mesh A and Mesh B is shown in Fig. 10c. Steps 1 and 2 are repeated for Mesh A to Mesh C pair and the Mesh B to Mesh C pair. The results for Steps 1 and 2 for these mesh pairs are shown in Figs 10d and 10e.

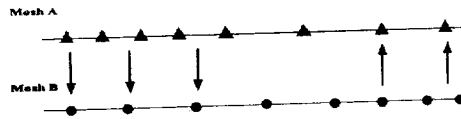
To complete the overlap optimization process, each point that is interpolated in a mesh is evaluated to determine which interpolation is to be kept. If only a single interpolation has been identified for a point, that interpolation is kept. If more than one interpolation has been identified (due to multiple mesh overlap), the interpolation with the smallest cdp is kept. Using this procedure, the resulting interpolations and field points (non-interpolated points) are shown in Fig. 10f. The field points in Fig. 10f show the effective optimized overlap that results from this approach. It also shows that Mesh C no longer has any active field points in this region because it is coarser than any of the other meshes with which it overlaps.



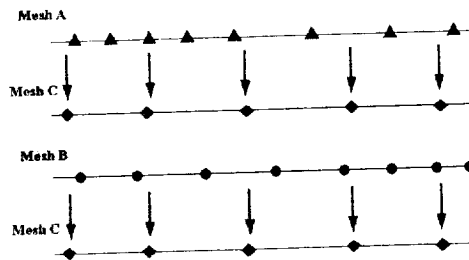
a. One Dimensional Meshes



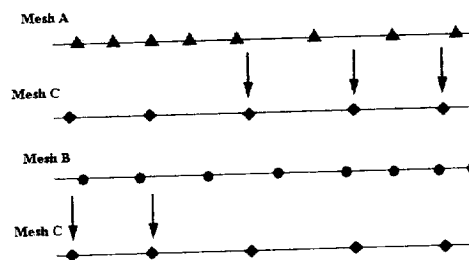
b. Step 1: Interpolate Between Meshes Keeping Only Coarser Mesh Points



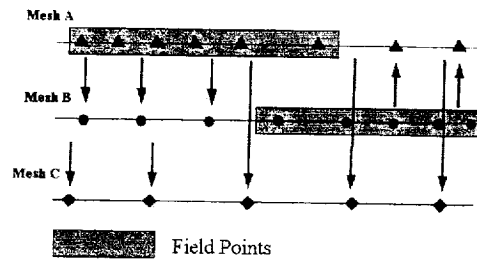
c. Step 2: Remove Interpolated Points That Are Part of a Donor Cell



d. Step 1 Repeated for Other Meshes



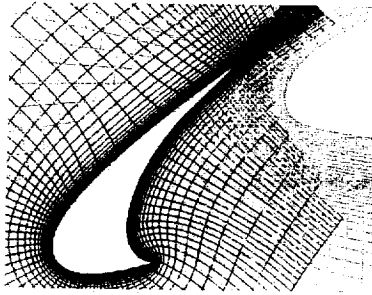
e. Step 2 Repeated for Other Meshes



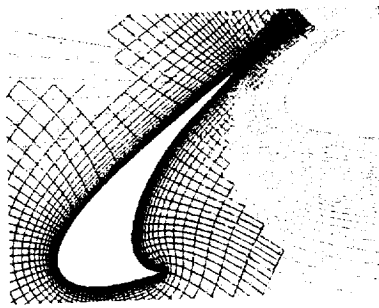
f. Step 3: Keep Finest Mesh Points

Fig. 10. Overlap Optimization Procedure

In Fig. 11, an example of two overlapping meshes and the resulting optimized overlap is shown. It can be seen that this procedure keeps the overlap region away from the tightly packed boundary layer region of both meshes. In Fig. 12, the resulting optimized overlap is shown for three meshes. The optimized overlap that is produced in this case would be very difficult to specify manually, and would be nearly impossible in three-dimensions.

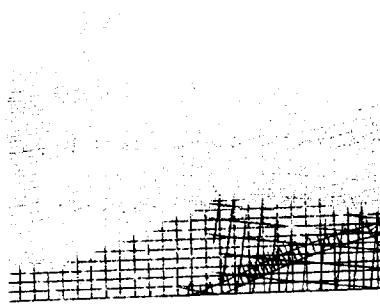


a. Non-optimized Overlap

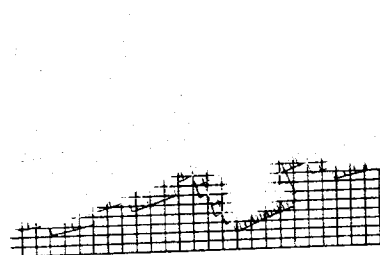


b. Optimized Overlap

Fig. 11. Flap/Airfoil Example



a. Non-optimized Overlap

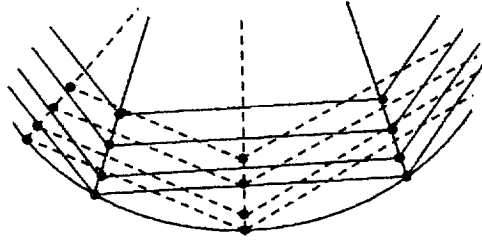


b. Optimized Overlap

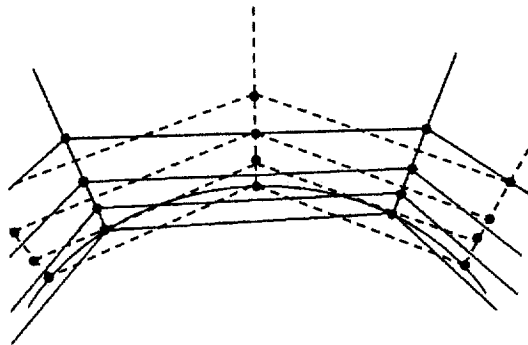
Fig. 12. Three Mesh Example

Projection for Viscous Grids

The oversetting process gives the user great flexibility in how the surface of a body is sub-divided into topologies that ease grid generation. As geometry has increased in complexity and the need for viscous solutions has increased, a problem with the overset approach for viscous grids has arisen. The problem, which is created by the linear discretization of curved surfaces, has manifested itself in two forms. These two forms are depicted in Figs. 13a and 13b, which depict two overlapping grids on a curved surface. The scale of these grids and the curvature of the surface is exaggerated in these figures to clarify the problem. The first problem type occurs for a concave surface (Fig. 13a). The surface points for both grids lie on the true surface of the body, but have points that do not have legal interpolation stencils. Therefore, any of these points that must be interpolated from the other mesh would be orphan points. This form of viscous interpolation problem is easily identified. The second form of viscous interpolation errors occurs for a convex surface (see Fig. 13b) and is not as easy to identify. In this form, the recipient points that require interpolation can find donor cells, but these donor cells are located much further away from the wall than the recipient points. Therefore, recipient points in the near-wall region of the boundary layer will receive data from cells in the outer region of the boundary layer. These viscous interpolation errors manifest themselves as large velocities near the surface. This error can lead to incorrect boundary layer profiles and significant errors in the flow solution.



a. Concave Surface



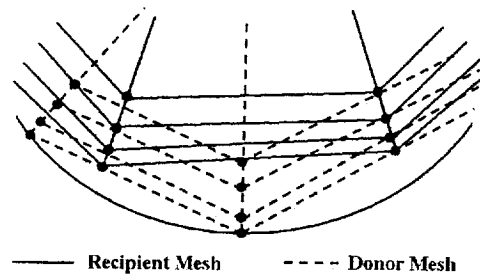
b.. Convex Surface

Fig. 13. Viscous Surface Interpolation Problems.

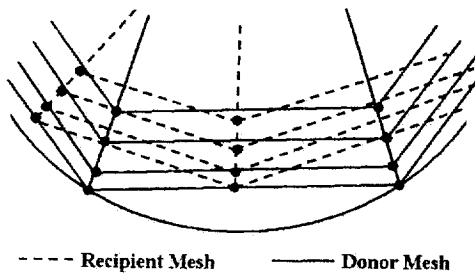
To correct this problem, the PROGRD code (Ref. 17) was developed. PROGRD is used to modify the grids prior to the interpolation process in PEGASUS. PROGRD, which projects one grid onto another, changes the final grids that are used in the flow solver. This approach leaves the user with a geometry that has been changed from the original, i.e., grid points that originally were on the solid surface are physically moved to faces of elements in a donor mesh.

The approach in PEGASUS 5 is to bring the PROGRD methodology into PEGASUS and project the grids, but only for determining the interpolation indices and coefficients. The original surfaces of the geometry are retained. The process is depicted in Fig. 14. The first step in this process is to project the recipient mesh onto the donor

mesh; points in the recipient mesh are then interpolated. The projected mesh is then discarded and only the interpolation indices and coefficients for the recipient mesh are retained. The process is then repeated for the mesh pair; this time with the identities of the donor and recipient meshes reversed. As before, only the interpolation indices and coefficients are retained. This process is repeated for all mesh pairs that have overlapping surfaces.



a. Projection of Recipient (Red) Mesh.



b. Projection of Recipient (Blue) Mesh

Fig. 14. Mesh Projection by Mesh Pair

Restarting

PEGASUS 5 is highly automated and will often yield excellent results with minimal input. However, there are occasions where some modification to the input is necessary to provide suitable communication among meshes in a complex chimera system. For example, a user may decide to modify a single grid in the system. A single grid will typically communicate only with a few other grids in the system. As a result, most of the information previously generated (i.e., interpolation coefficients between the other grids) are valid, and should not have to be repeated.

PEGASUS 5 employs a unique restarting capability whereby only processes that involve dependences on the modified input are repeated. PEGASUS 5 automatically determines what work needs to be performed to complete a restart execution. In this manner, a user can refine the PEGASUS 5 solution incrementally and inexpensively, rather than by repeating the entire solution for each input modification. In fact, restarting in PEGASUS 5 is very similar to using the UNIX "make" utility, in that previously performed processes that are independent of local changes are not repeated.

Parallelization

A parallel version of the PEGASUS software was developed using the Message Passing Interface (MPI) standard. The architecture of the PEGASUS 5 software was designed from the very beginning to be very amenable to coarse-grained parallelization. Nearly all of the computations done in the code consist of a number of operations using data from either an individual mesh or pairs of meshes. These operations include surface projections between all mesh pairs, building alternating-digital trees (ADTs) for each mesh, interpolation stencil searches between all mesh pairs, hole-cutting operations on individual meshes, and boundary point identification on individual meshes. Most of these operations are independent of each other and can be performed simultaneously. However, there are some processes that are required to be handled sequentially with respect to each other, e.g., all projection operations must precede all of the interpolation operations. The parallelization was implemented by creating a single master process, and $NP-1$ worker processes, where NP is the number of MPI processes assigned to the job. The master initializes the entire PEGASUS execution, and then asynchronously assigns individual operations to each of the workers. Once a worker reports back to the master that it has completed its operation, the master sends it a new operation to perform. Figure 15 shows a graphical representation of the operations being

performed during an actual PEGASUS 5 execution by each of the workers as a function of time, where a total of 15 processors (14 workers) were used. A close-up view of some of the operations is shown in Fig. 16. This shows worker processors computing the ADT operations, the interpolation operations, and an automatic-hole boundary operation. Notice that most workers become idle for a brief time waiting for the last ADT operation to complete before the interpolation operations are initiated.

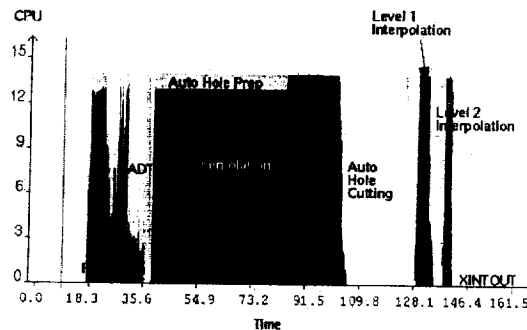


Fig.. 15. PEGASUS 5 Operations versus Time, (Harrier Jet Grid System)

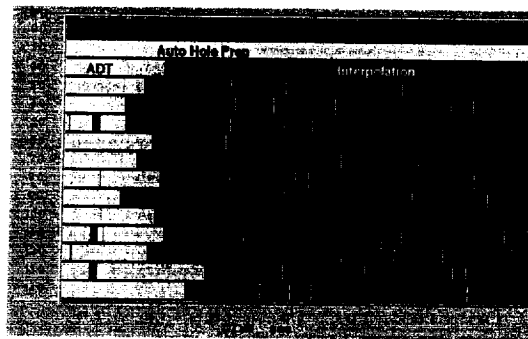


Fig.. 16. Close-up View of the Operations being Performed by Worker Processes

The test case used in Figs 15 and 16 was the processing of a grid system for a Harrier jet in a hover mode; this grid system was used in the computational study by Chaderjian et al.(Ref. 18). The Harrier grid system contains 52 meshes, and the PEGASUS 5 execution required 2429 separate operations. Figure 17 shows the parallel performance of PEGASUS 5 running the Harrier problems on both an SGI Origin O2K and Origin O3K at NASA Ames Research Center (named "Lomax" and "Chapman" respectively). The straight dashed line in the figure

represents the theoretical maximum speed up if all NP-1 workers are kept busy all of the time. Good parallel performance is seen up to about 16 processors, after which the parallel speedup asymptotes at a maximum of a factor of 14. The asymptotic behavior of the parallel speedup is expected for this coarse-grained approach. The final process, labeled as the XINTOUT process in Fig. 15, is performed in serial in the current approach. This operation is the last step in the code in which all the final interpolation stencils are gathered and written to the final output file.

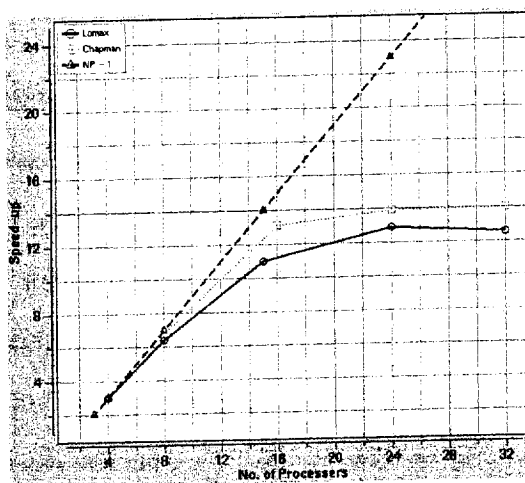


Fig. 17. Parallel Performance for the Harrier Jet Configuration on SGI Origin O2K and O3K Systems

One need not run PEGASUS 5 on a large parallel system to take advantage of the parallelization. For instance, a user working on a dual-processor workstation can utilize this capability to reduce the PEGASUS 5 execution turn-around by a factor of two. This enables a user to effectively perform overset grid connectivity for a complex three-dimensional problem in one day, which represents at least order of magnitude improvement over the use of the old PEGASUS 4 code. The earlier version would often require on the order of 10 to 20 days to perform such an operation, (see Ref. 19), and requires significantly greater user expertise than is required by PEGASUS 5. The realization of a production-ready code to automatically perform the overset pre-processing represents a major step toward realizing the 1996 strategic goal of the NASA/Boeing Advanced Subsonic (AST) Program of reducing cycle time for a complex 3D problems from hundreds of days to 5 days (see Ref. 19).

Examples

Three example PEGASUS 5 test cases are presented. They are a three–element high–lift airfoil, a generic business jet configuration, and a Boeing 777 aircraft in a landing configuration. The user modifications to the input and the required computing resources are described for each case. For the first and last cases, flow solutions have been computed using the results of both PEGASUS 4 and PEGASUS 5 connectivity files. These flow computations utilized the OVERFLOW (Ref. 6 and 20) flow solver.

Two types of input files are required by the PEGASUS 5 software: the volume grid files, and a text input file. The grid files contain the coordinates for each individual mesh, and the text file contains the boundary conditions, and possibly other input variables to customize the behavior of the software. There are several utility programs that come with the PEGASUS 5 software to aid in the generation of these input files. If one is preparing a grid system for use by the OVERFLOW flow solver, the user need only supply the OVERFLOW input file (containing all of the boundary conditions), and a composite grid file containing the coordinates of all of the individual meshes. A script ("peg_setup") reads these two files and generates all of the input files required by PEGASUS 5. This method was used for each of these examples.

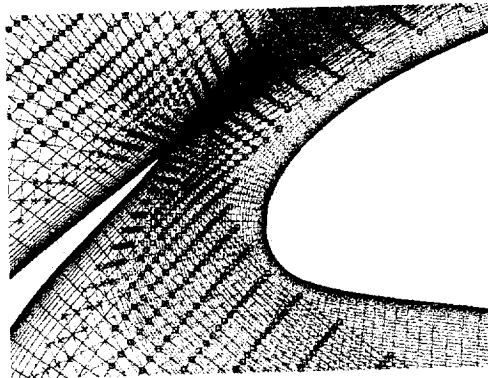
A description of all of the possible input variables recognized by PEGASUS 5 is beyond the scope of this paper. The most commonly used input variable which a user might need to modify to fix problems with the automatic hole cutting operation is the OFFSET input variable. This variable may be specified globally for all meshes at once, or independently for each individual mesh. The default value of OFFSET is zero. Values greater than zero cause the code to enlarge any holes in a particular mesh. It does this by examining every point in a mesh; if a point is within OFFSET cells of a hole point, then it too gets blanked.

Multi–Element Airfoil

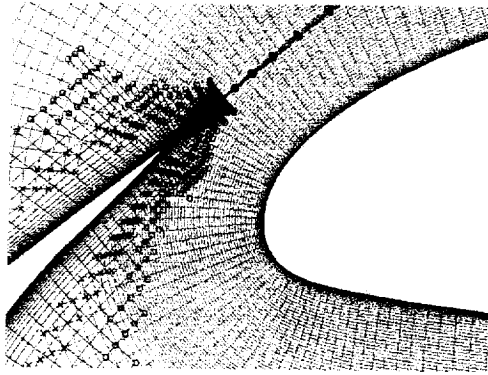
The first test case is a two–dimensional three–element airfoil known as the 30P30N configuration (Ref. 21), which was built and tested extensively by the former McDonnell–Douglas company and NASA Langley Research Center. It has been commonly used as a high–lift CFD validation configuration. The airfoil consists of a main

wing section with a leading-edge slat and a trailing-edge flap. The grid system consists of seven zones and 313,000 grid points. These overset grids were used previously as a test case for an automated grid-generation procedure (Ref. 22) which used the PEGASUS 4 software.

The only modification to the automatically generated PEGASUS 5 input file was to increase the global OFFSET variable to 2, and to set the OFFSET variable to a value of 5 for both of the "box" grids. PEGASUS 5 ran in 65 seconds on a single SGI R10000 250Mhz CPU. Details of the resulting grid system are shown in Figs 18 and 19. Figure 18 depicts the slat and wing grids at the slat trailing edge and wing leading edge. The symbols indicate the location of fringe points, which are boundary points in a grid that will receive interpolation boundary conditions from its overlapping neighbor. Figure 18a shows all of the fringe points; Fig. 18b shows only fringe points which are one or two points away from an active interior point. Thus Fig. 18a shows the actual hole cut by PEGASUS 5; Fig. 18b shows the effective hole. Note that the actual hole in the wing grid is too close to the slat trailing edge. This is because the Cartesian hole-map typically does not have enough resolution to resolve an extremely thin trailing edge. However, as can be seen in Fig. 18b, the effect of the interpolation optimization is to create a larger hole in the wing near the slat trailing edge.



a. All Fringe Points

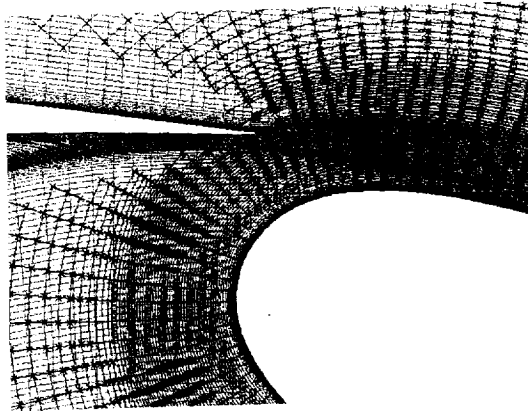


b. First and Second Fringe Points

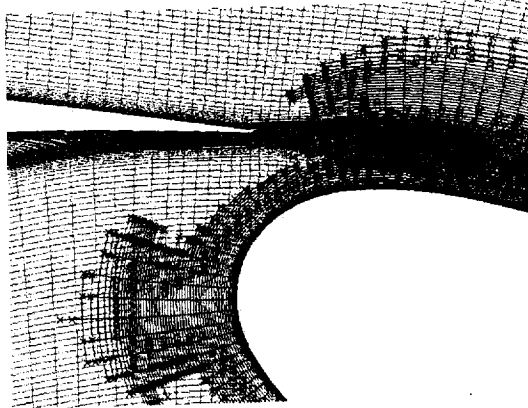
Fig. 18. Fringe Points Near the Slat Trailing Edge

Figure 19 shows the grids around the wing and the flap in the vicinity of the wing trailing edge and the flap leading edge. Again the symbols mark the fringe points. Figure 19a shows the actual hole, and Fig. 19b shows the effective hole.

The resulting grid system was used to compute a solution with the OVERFLOW flow solver. A flow solution was also computed for a grid system obtained by running the PEGASUS 4 code with the same meshes used by PEGASUS 5. For these computations the free-stream Mach number was 0.2, the Reynolds number was 9 million, and the angle of attack was 8.1 degrees. Very similar results were obtained for both grid systems. Figure 20 plots the pressure coefficient results for these calculations together with some experimental results for this geometry. The experimental values are plotted with the circles, the PEGASUS 4 results are plotted with a dashed line, and the PEGASUS 5 results are plotted with a solid line. Although the CFD results have suction peaks that are higher than the experimental results, there is no visible difference between the PEGASUS 5 and PEGASUS 4 results.



a. All Fringe Points



b. First and Second Fringe Points

Fig. 19. Fringe Points Near the Wing Trailing Edge

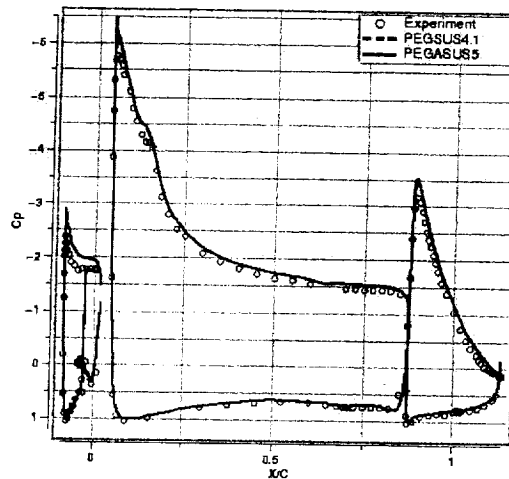


Fig. 20. Pressure Coefficient Results on Three-Element Airfoil

Business Jet

This test case is a generic business jet geometry consisting of a fuselage, wing, and an aft-mounted pylon and nacelle. The grid system consists of 13 meshes and 3.5 million grid points. Two modifications were made to the automatically generated input file in order to produce a high-quality grid system. The first of these changes was to increase the global value of the OFFSET parameter from 0 to 1, and to increase the OFFSET value to 2 for the nacelle mesh. This improved the automatically-generated holes and blanked some points that were left inside the thin trailing edges of the wing, the pylon, and the nacelle during the initial run. The second change was the "unblanking" of an automatic hole that was being cut into the surface grid of the wing fillet. This type of undesirable automatic hole cut can occur where two or more surface grids overlap in a region with significant surface curvature. When this occurs it can usually be fixed very easily by specifying a range of grid indices in the input file where PEGASUS 5 is not allowed to cut any holes.

With the addition of these changes to the input file, PEGASUS 5 produces a good-quality grid system for this case. The code required 30 minutes of CPU time on a single SGI R10000 250Mhz CPU for the initial run using the automatically generated input file. After making the above changes to the input file, the restart execution required only 2.5 CPU minutes. The resulting grid was left with only 19 orphan points, i.e., boundary or fringe

points for which the code was unable to find an acceptable interpolation donor cell from a neighboring mesh. A small number of orphan points is considered acceptable by practitioners of overset CFD methods. Most flow solvers, such as OVERFLOW, have a procedure to update the dependent variables at ORPHAN points by averaging the dependent variables from adjacent computed grid points in the same mesh.

Figure 21 shows the fringe and boundary interpolation points in the symmetry plane, where they are marked with circles. The rectangular-shaped grid cells are part of an inner-box mesh that surrounds the body-fitted meshes. The grid system also contains an outer box mesh that extends to the far-field boundary which is not shown in the figure. The figure shows the double-row of outer boundary points belonging to the inner box mesh which receive interpolated information from the outer box mesh. The darker symbols surrounding the airplane show the outer-boundary points of the fuselage mesh which receive interpolation from the inner box mesh. The lighter symbols show the fringe points surrounding the hole cut in inner box mesh; these receive interpolated information from the fuselage mesh. Figure 22 shows grid planes in a constant stream-wise plane which intersect the center of the nacelle and pylon. This figure also shows the symbols denoting all of the fringe and outer boundary interpolation points in these grid planes. This figure shows the complex intersections of the nacelle grid, the two pylon grids, the fuselage grid, and a box grid which surrounds the pylon and nacelle.

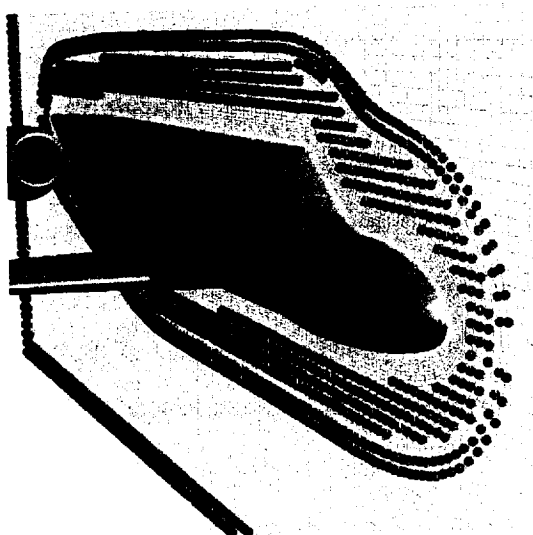


Fig. 21. Business Jet Fringe Points in the Symmetry Plane

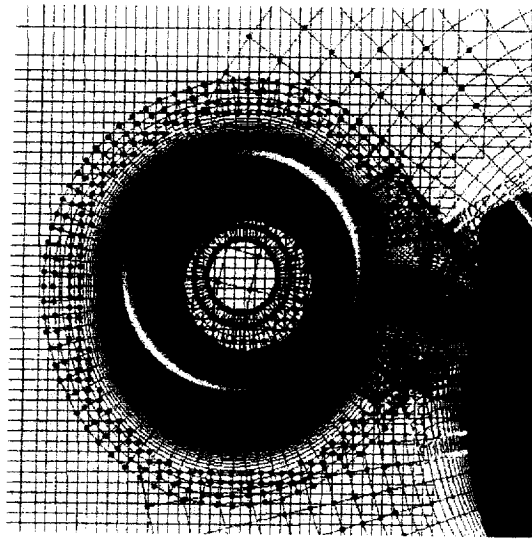


Fig. 22. Business jet Fringe Points Surrounding the Nacelle

Boeing 777 Aircraft

The final example is by far the most complex geometry used to test PEGASUS 5: a Boeing 777-200 aircraft in a landing configuration with 30-deg flap deflection. Figure 23 shows the surface grids used for this case, with only every fourth grid point plotted in each direction for clarity. The geometry includes the fuselage, vertical tail, wing, pylon, flow-through nacelle and core cowl, inboard and outboard slats, a leading-edge krueger slat, double-slotted inboard flaps, flaperon, and outboard flap, and the three largest flap-hinge fairings. Figure 24 shows the view of the surface grids from underneath the aircraft, looking inboard at the flaps and the inboard flap-hinge fairing. The grid system for this geometry was originally developed by Rogers et al. (Ref. 23) as part of the NASA and Boeing Advanced Subsonics Technology Program. This geometry was used as a demonstration case to meet a program milestone requiring a complete high-lift aircraft CFD simulation to be performed in 50 labor days. The milestone was met by computing the first solution, with just the CAD definition as the initial starting point, with 48 labor days of effort. Of this time, 32 labor days were required to perform the over-setting of the volume grids using the PEGASUS 4 software.

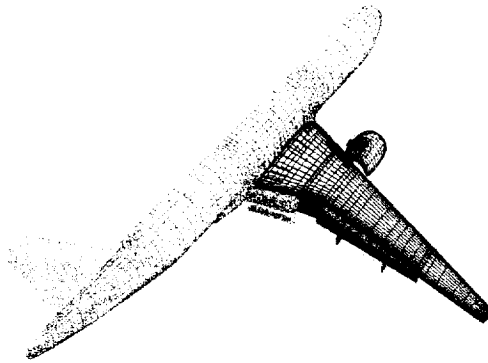


Fig. 23. Surface Grids on Boeing 777-200

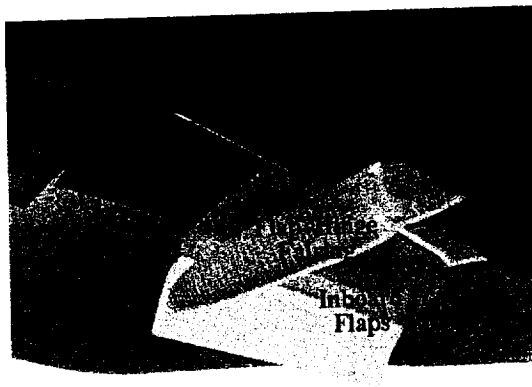


Fig. 24. Underside View of Flaps and Flap-Hinge Fairing

The 777 volume grids consist of 79 meshes and 22.4 million grid points. For this problem, PEGASUS 5 initially was executed on 16 processors of an SGI Origin O2K system. The code required 40 minutes to run from start to finish, in which just over 9 CPU hours of execution time was accumulated, for a parallel efficiency of 85%. After the initial run of the code, it was apparent that the default automatic hole cutting did not have nearly enough resolution for the wide range of length scales in this problem. In particular, the small gaps between the high-lift elements are about three orders of magnitude smaller than the fuselage length and the wing semi-span. Thus the first modification made to the default PEGASUS 5 input was to create a number of additional automatic Cartesian-hole maps. Each of the high-lift elements which formed a fully-enclosed surface were used to create a separate hole cutter; this included the two slat elements and three of the four flap elements. The hole-cutter

composed of the fuselage and wing was split spanwise into three separate hole cutters. This was quite easy to accomplish by manually specifying the minimum and maximum coordinates for each cutter. Each of these three hole-cutters was also given 50% increased resolution in both longitudinal and vertical directions. After re-running PEGASUS 5 with these eight automatic hole-cutters, the input was further refined by increasing the OFFSET value to 1 or 2 for nearly half of the meshes. It was also found that a large number of orphan points were created because the default limits on the surface-to-surface projections were too restrictive. The maximum allowable projection distance was increased by 50% to fix this problem. Finally, two regions of some overlapping grids near the surface had to be "unblanked" to correct for some bad hole cutting through the surfaces of some overlapping grids.

After these input-file modifications, a final grid system was obtained which contained just under 1200 orphan points. This compared very favorably to the grid system created by PEGASUS 4, which had just under 5600 orphan points. The total labor time spent running and modifying the PEGASUS 5 inputs was three days, an order of magnitude decrease of the 32 days required by PEGASUS 4 for these same volume grids. Furthermore, the input modifications required for PEGASUS 5 were significantly simpler compared to the user input required by PEGASUS 4.

Subsequent runs of PEGASUS 5 for the 777 grid system were performed to test the parallel performance of the code. The parallel speed-up for the 777 was better than for the Harrier grid system, as it was able to benefit from the use of more processors. The 777 grids were run on 48 SGI Origin processors, which provided a speed-up of a factor of 33 over the use of a single processor. Using 48 processors, the code was able to process the 777 grids in less than 13 minutes.

The OVERFLOW code was run using this new PEGASUS 5 grid system in order to compare with the computed flow results from the PEGASUS 4 grid system. Several angles of attack were run, matching the PEGASUS 4 cases reported in Ref. 23. The Mach number was 0.2, and the Reynolds number based on the mean aerodynamic chord was 5.8 million. The lift coefficients for these new runs are plotted in Fig. 25, together with the previous computational results and experimental results for this geometry.

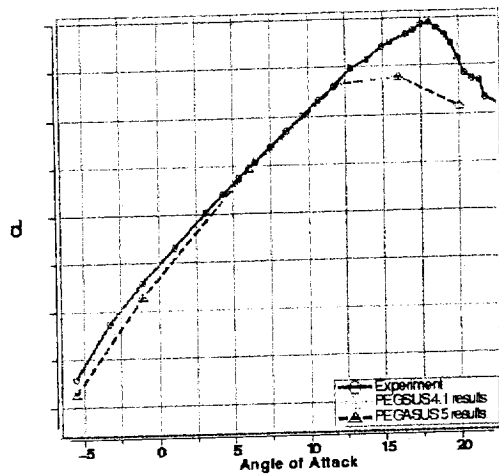


Fig. 25. Lift Coefficient versus Angle of Attack, Comparing OVERFLOW Results for the PEGASUS 4 and PEGASUS 5 Grid Systems

The actual value of the lift coefficient is not included on the vertical axis labels due to the proprietary nature of this data. It can be seen that the PEGASUS 5 results match very well with the PEGASUS 4 computations. The new results show a slight decrease in lift at the negative angles of attack, further away from the experimental data, and a slight increase in lift at the highest angles of attack, where the computed solution has stalled over the inboard portion of the wing. Thus the computational results fail to predict maximum lift; a discussion of the possible reasons for this is given in Ref. 23. However, the computational results do agree well with the experimental data at angles of attack of 12 degrees and lower, and at typical approach conditions, the computed lift is within 1.5% of the experimental lift.

Conclusion

The newest version of PEGASUS, version 5, has been automated to reduce the number of user inputs and the time required to determine the inter-connectivity between overlapping meshes. Automation of the hole cutting and outer boundary specification is based on the inputs required by the flow solver, which can be automatically generated by other readily available overset-CFD software. This greatly decreases the user input requirements.

Additionally, the overlap optimization and viscous interpolation projection improve the inter-connectivity solutions that are produced by PEGASUS 5.

The modular design of the PEGASUS 5 software made it straightforward to implement a coarse-grain parallel approach using the MPI message passing library. The parallel version of the code will always reproduce the same results as the serial version. It exhibits efficient execution speed-up for a modest number of processors, depending on the problem size. Over factor of 33 speed-up on 48 processors was obtained for the Boeing 777-200 test case.

The computed OVERFLOW results illustrate that grid systems produced by the new version of PEGASUS lead to the same results as those from the old version, but at a significant cost savings in terms of both effort and required user expertise. The amount of user time and expertise required for the Boeing 777-200 aircraft was an order of magnitude less than that required by the PEGASUS 4 code for processing the same volume grids.

References

1. Benek, J.A., Dougherty, F.C., and Buning, P.G., "Chimera: A Grid-Embedding Technique," AEDC-TR-85-64, December 1985.
2. Meakin, R.L. "Object X-rays for Cutting Holes in Composite Overset Structured Grids," AIAA Paper 2001-2537, June 2001.
3. Belk, D.M. and Maple, R.C., "Automated Assembly of Structured Grids for Moving Body Problems," AIAA 95-1680, June 1995.
4. Wang, Z.J., Parthasarathy, V., and Hariharan, N., "A Fully Automated Chimera Methodology for Multiple Moving Body Problems," AIAA 98-0217, January 1998.
5. Brown, D.L., Henshaw, W.D., and Quinlan, D.J., "Overture: Object-Oriented Tools for Overset Grid Applications," AIAA conference on Applied Aerodynamics, UCRL-JC-134018, 1999.
6. Buning, P. G., Jespersen, D. C., Pulliam, T. H., Chan, W. M., Slotnick, J. P., Krist, S. E., Renze, K. J., "OVERFLOW User's Manual, Version 1.8b," NASA Langley Research Center, Hampton VA, 1998.

7. Tramel, R., and Nichols, R., "A Highly Efficient Numerical Method for Overset-Mesh Moving-Body Problems," AIAA Paper No. 97-2040, June 1997.
8. Rogers, S.E., Kwak, D., and Kiris, C., "Numerical Solution of the Incompressible Navier-Stokes Equations for Steady-State and Time-Dependent Problems," AIAA Paper 89-0463, Jan. 1989. Published in AIAA Journal, Vol. 29, No. 4, Apr. 1991, pp. 603--610.
9. Lijewski, L.E., and Suhs, N.E., "Time-Accurate Computational Fluid Dynamics Approach to Transonic Store Separation Trajectory Prediction," Journal of Aircraft, Vol. 31, No.4, pp.886-891, Jul.-Aug. 1994.
10. Benek, J.A., Donegan, T.L. and Suhs, N.E., "Extending the Chimera Grid Embedding Scheme with Applications to Viscous Flow." AIAA Paper No. 87-1126, June 1987.
11. Dietz, W.E. and Suhs, N.E., "PEGSUS 3.0 Users Manual." AEDC-TR-89-7, August 1989.
12. Suhs, N.E. and Tramel, R.W., "PEGSUS 4.0 Users Manual," AEDC-TR-91-8, November 1991.
13. Slotnick, J. P., An, M. Y., Mysko, S. J., Yeh, D. T., Rogers, S. E., Roth, K. R., Nash, S. M., and Baker, M. D., "Navier-Stokes Analysis of a High-Wing Transport High-Lift Configuration With Externally Blown Flaps," AIAA Paper 2000-4219, Aug. 2000.
14. Chan, W., "The OVERGRID Interface for Computational Simulations on Overset Grids," AIAA Paper 2002-3188, June 2002.
15. Chan, W. M., Rogers, S. E., Nash, S. M., Buning, P. G. and Meakin, R. L., "User's Manual for Chimera Grid Tools," Version 1.6, <http://www.nas.nasa.gov/~rogers/cgt/doc/man.html>, NASA Ames Research Center, Oct. 2001.
16. Bonet, J. and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," International J. Numerical Methods in Engineering, Vol. 31, 1991, pp. 1--17.
17. Chan, W., Buning, P. G., and Krist, S. E., "PROGRD User's Manual, Version 0.5," NASA Ames Research Center, Moffett Field, CA, Aug. 1998.
18. Chaderjian, N. M., Pandya, S., Ahmad, J., and Murman, S. M., "Parametric Time-Dependent Navier-Stokes Computations for a YAV-8B Harrier in Ground Effect," AIAA Paper No. 2002-0950, January 2002.
19. Rogers, S. E., Roth, K., Nash, S. M., Baker, M. D., Slotnick, J. P., Whitlock, M., and Cao, H. V., "Advances in Overset CFD Processes Applied to Subsonic High-Lift Aircraft," AIAA Paper 2000-4216, August, 2000.

20. Jespersen, D. C., Pulliam, T. H., and Buning, P. G., "Recent Enhancements to OVERFLOW," AIAA Paper 97-0644, Jan. 1997.
21. Chin, V., Peters, D., Spaid, F., and McGhee, R., "Flowfield Measurements about a Multi-Element Airfoil at High Reynolds Numbers," AIAA Paper 93-3137, July 1993.
22. Rogers, S. E., Cao, H. V., and Su, T. Y., "Grid Generation For Complex High-Lift Configurations," AIAA Paper 98-3011, June 1998.
23. Rogers, S. E., Roth, K. R., Cao, H. V., Slotnick, J. P., Whitlock, M., Nash, S. M., and Baker, M. D., "Computation Of Viscous Flow For A Boeing 777 Aircraft In Landing Configuration," AIAA Paper 2000-4221, Aug. 2000. Published in Journal of Aircraft, Vol. 38, No. 6, Dec. 2001, pp. 1060-1068.