AIAA JOURNAL
Vol. 41, No. 6, June 2003

PEGASUS 5: An Automated Preprocessor for Overset-Grid Computational Fluid Dynamics

Stuart E. Rogers*
NASA Ames Research Center, Moffett Field, California 94035
Norman E. Suhs†
U.S. Army, Redstone Arsenal, Alabama 35898
and
William E. Dietz‡
Flow Analysis, Inc., Tullahoma, Tennessee 37330

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 small discretization errors in overset surfaces, more efficient interpolation search methods using an alternating digital tree and a stencil-jumping scheme, 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 speedup of the execution time by an order of magnitude, and up to a factor of 30 for very large problems. The results of two example cases are presented: a three-element high-lift airfoil and a complete Boeing 777-200 aircraft in a high-lift landing configuration. Comparisons of the computed flowfields for these 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 interconnectivity among the grids. The initial creators of the chimera grid-embedding scheme[1] developed the first versions of the PEGASUS code[2] to establish the data structure required for communication among the overlapping structured meshes. The output from PEGASUS is the interpolation data that are passed to the flow solver; these data include 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, that is, blanked, from the computational domain because they are interior to a solid body. These points are also known as hole points.

There are several codes that perform the same basic functions as PEGASUS. These include DCF3D,[3] Beggar,[4] FASTRAN,[5] and Overture.[6] 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,[7,8] NXAIR,[9] INS3D,[10] and a cell-centered code, DXEAGLE.[11]

Received 6 May 2002; presented as Paper 2002-3186 at the AIAA 32nd Fluids Conference, St. Louis, MO, 24–26 June 2002; revision received 6 January 2003; accepted for publication 6 January 2003.

*Aerospace Engineer, Advanced Supercomputing Division, Mail Stop T27B-1. Associate Fellow AIAA.
†Aerospace Engineer, Aeromechanic Division, Aviation Engineering Directorate, Aviation and Missile Command, AMSAM-RD-AE-A (Norman Suhs), Building 4488, B-Wing. Senior Member AIAA.
‡Senior Research Scientist, 351 Woodland Way. Member AIAA.

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. The following three subsections contain an overview of the chimera approach, a brief history of the PEGASUS code, and then the motivation for the development of the newest version of the software.

Overview of Chimera Approach

The interpolation process is further illustrated in Fig. 1, which depicts a portion of the overlap region between two meshes. Mesh 1 points that are inside the solid body of mesh 2 are excluded from the computational domain. In chimera terminology, these points are blanked out and are then known as hole points. The points in mesh 1 surrounding the blanked points are known as hole-fringe points; they receive flowfield information interpolated from mesh cells within mesh 2. These hole-fringe points are denoted with square symbols in Fig. 1. Correspondingly, points on the outer boundary of mesh 2 receive flowfield information interpolated from cells in mesh 1. These points are denoted by the circular symbols in Fig. 1. Generally, any mesh can receive information from any other overlapping mesh to update its outer-boundary and hole-fringe points.

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[2] had limited connectivity and hole cutting capabilities. In particular, a grid hierarchy was imposed, that is, 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.

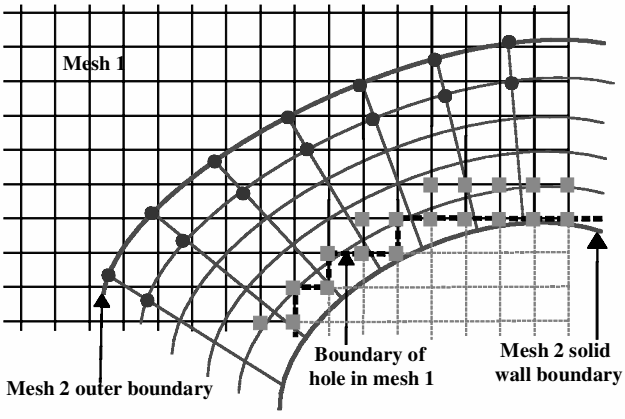


Fig. 1 Detailed view of overlap region.

Version 2 of PEGASUS[12] 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[13] 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[14] 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[15] 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, two 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. 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, as were illustrated in Fig. 1. 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 that lies on the boundary of a computational mesh and that 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 preceding step. If a suitable donor cell cannot be found for an interpolation point, the point is termed an "orphan."

In general, orphan points need to be eliminated. If any are left over after running the PEGASUS code, the user must investigate the cause and then modify the code input or the volume grids. Some flow solvers do have a method for updating orphan points through an averaging process because it has been found that, in certain situations, orphan points cannot be eliminated. A small number of orphan points in a noncritical region of the flowfield will have only a negligable effect on the solution accuracy. However, the best standard is to eliminate orphan points entirely.

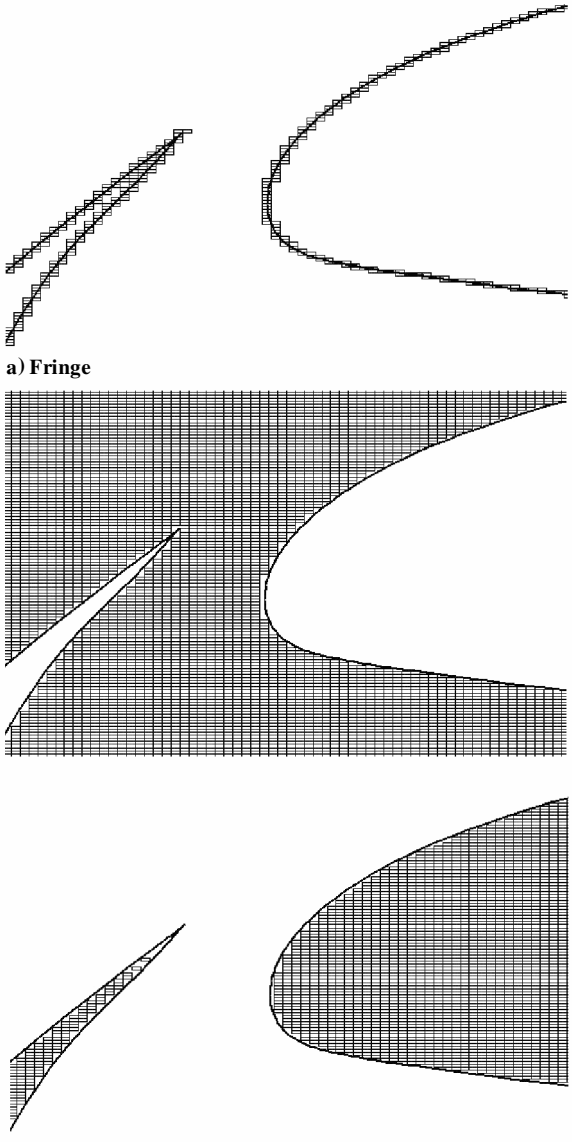
The first two of the steps just described require knowledge of the complete set of boundary conditions that are to be applied by the flow solver for each mesh. The PEGASUS 5 code, therefore, requires the flow-solver boundary conditions as part of its input. The format and definitions of the boundary conditions for this input were adopted from the OVERFLOW code. Some grid-generation software provides a mechanism to produce an OVERFLOW input file automatically. For example, the OVERGRID software, which is part of the Chimera Grid Tools package,[16] 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 backward compatibility. The automated hole cutting approach is based on a Cartesian hole-map coupled with a line-of-sight algorithm. Parts of the current approach were based on ideas originally developed by J. Steger (private communication) and by Chiu and Meakin.[17] The automatic hole generation process is illustrated using a two-dimensional three-element airfoil example. The current hole cutting procedure requires that 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.

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 that fully encompasses the solid boundaries of the configuration. The elements of the Cartesian mesh that intercept the solid surface of the airfoil are identified and designated as fringe elements. Some of the fringe elements in the slat-wing region of the three-element airfoil are depicted in the top of Fig. 2.

It is assumed that the corner elements of the Cartesian mesh are outside elements. Any unidentified, that is, nonfringe, element that is adjacent to an outside element must itself be an outside element.



a) Fringe

c) Inside

Fig. 2 Elements.

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. 2b. Finally, any element remaining that is not either an outside or fringe element must be an inside element. Inside elements are drawn in Fig. 2c.

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 whether a clear line of sight exists between the point and an outside or inside, that is, nonfringe, element; 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 nonfringe element does not intersect the solid surface contained in the fringe element. This algorithm is illustrated in Fig. 3. Points that can "see" an outside element are field points; points that can see an inside element are hole points. In the example of Fig. 3, point A is outside. Point B is inside and will be marked as a hole point.

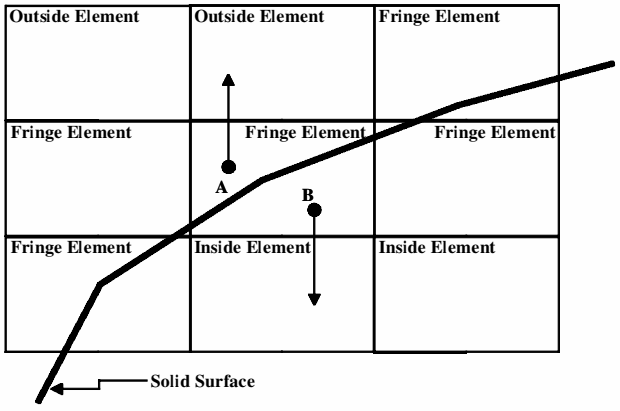


Fig. 3 Line-of-sight algorithm.

Outer-Boundary Specification

The automation of the outer-boundary point specification is straightforward because 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." For example, the symbols in Fig. 1 illustrate a fringe level of two. 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. Note 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 subprocesses, 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 subprocesses: 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 $N \times (N-1)$ possible donor/recipient grid pairs, where N is the number of grids in the configuration. Because 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 subprocess 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 subprocess next loops through every single grid point in the recipient grid. Inside this loop, it first tests to see whether 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).[18] 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 has been identified, a stencil-jumping algorithm is used to find the donor cell that contains the recipient

point. Given the target-point x, y, and z coordinates and the current close donor cell, the stencil-jumping process inverts the equations for trilinear interpolation using a Newton iteration. This solves for the three computational-space indices relative to the current donor cell. If these indices are between 0.0 and 1.0, the target point is contained in the current donor cell and the process terminates. Otherwise, these indices are used to provide a direction and distance to jump from the current cell to a new donor cell, and the process is repeated. The Newton iteration generally requires only from three to five inversions, and only two or three stencil jumps are required if the initial donor cell is close enough and the grid is smooth.

This stencil-jumping algorithm and its variants are the most commonly used approaches for finding donor cells. This algorithm has also been called "gradient search" and "stencil walking." The difference between the stencil-walking and stencil-jumping approaches is that the former search is restricted to move one grid cell at a time, whereas the latter allows the search to jump across multiple grid cells in one step. See Refs. 3 and 19 for more details on the stencil-walking algorithm, including the equations for trilinear interpolation and the Newton scheme used to solve them.

The implementation of the stencil-jumping procedure includes some additional enhancements to improve its robustness. The stencil jump is limited to a maximum of five grid cells per jump in any one computational direction. Another 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 for a particular boundary or fringe recipient point to have two or more possible donor cells. A new algorithm has been implemented in PEGASUS 5 that 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 sizes of the donor and recipient cells in the overlapping region differ significantly. 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

$$\mathrm{CDP} = \sum_{j=1}^{3} \frac{(X_j)_{\mathrm{DB}} \times V_B - (X_j)_{\mathrm{DI}} \times V_I}{(X_j)_{\mathrm{DB}} \times V_B}$$

where $(X_j)_{\mathrm{DB}}$ is the maximum of the jth component of the four diagonals of the boundary cell, $(X_j)_{\mathrm{DI}}$ is the maximum of the jth component of the four diagonals of the interpolation cell, V_B is the volume of the minimum Cartesian cell encompassing the boundary cell, and V_I is the volume of the minimum Cartesian cell encompassing the interpolated cell. Values for the cell difference parameter will vary from 0 (the best) to very large values.

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. Other authors have also attempted to resize the holes and optimize the grid overlap automatically[17,20]; however, the current approach is unique. The 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 whereas the coarser mesh points should be interpolated from the finer mesh points. To demonstrate the steps used to achieve the overlap optimization, three one-dimensional meshes are used (Fig. 4a).

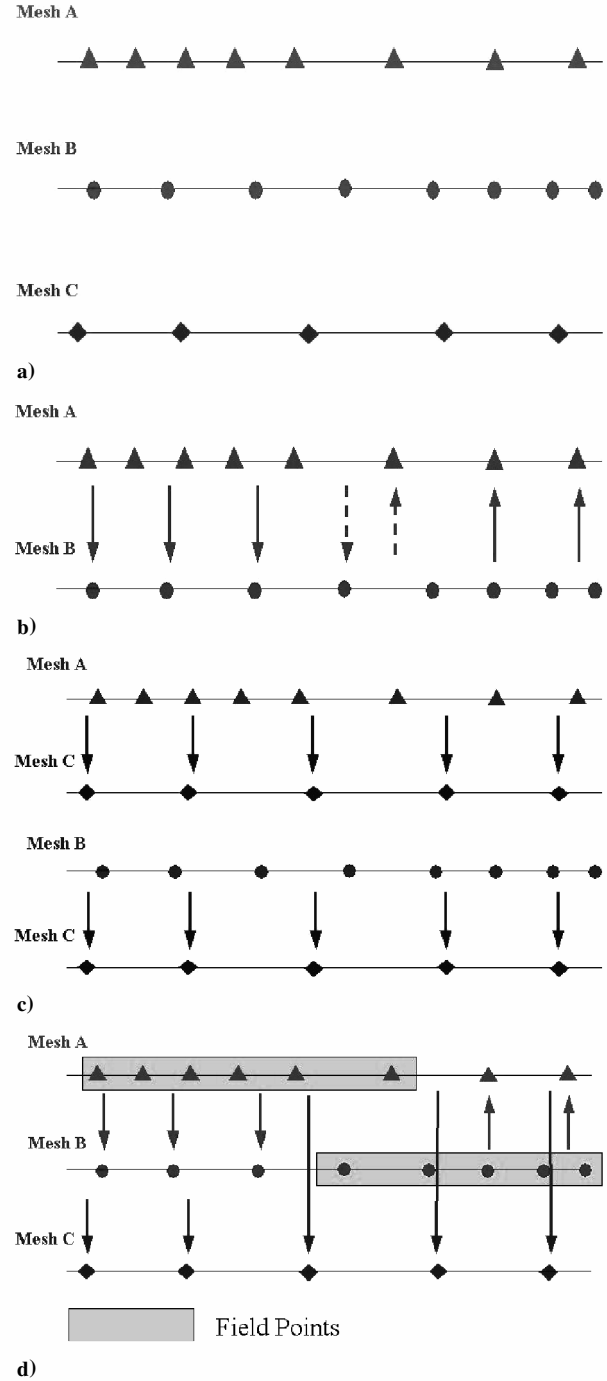
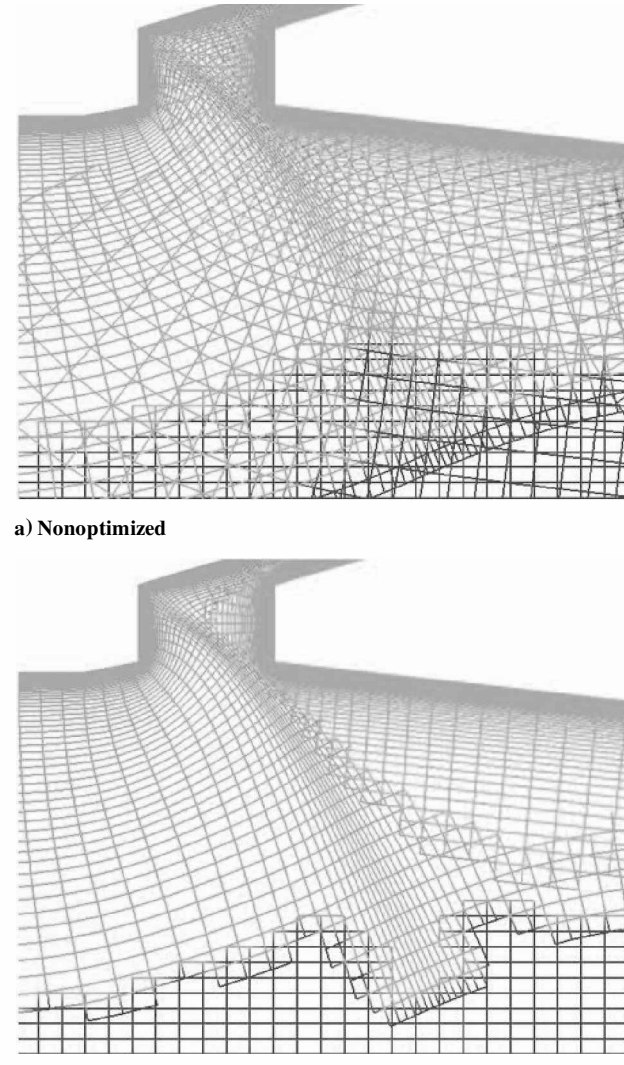


Fig. 4 Overlap optimization procedure: a) one-dimensional meshes; b) step 1, interpolate between meshes keeping only coarser mesh points; c) steps 1 and 2 repeated for other meshes; and d) step 3, keep finest mesh points.

Mesh A is stretched from fine to coarse, mesh B is stretched from coarse to fine, and mesh C has constant spacing that is coarser than both meshes A and 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 (Fig. 4b). The arrows in Fig. 4 and the remainder of Fig. 4 indicate the direction of the data flow. The head of the arrow points to the interpolated point, whereas 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 whether 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 meshes A and B in this step is to remove the interpolation indicated by the dashed arrows in Fig. 4b. Steps 1 and 2 are repeated for the meshes A–C pair and the meshes B–C pair. The result for steps 1 and 2 for these mesh pairs is shown in Fig. 4c.

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. With this procedure, the resulting interpolations and field points (noninterpolated points) are shown in Fig. 4d. The field points in Fig. 4d 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. In Fig. 5, an example of three overlapping meshes and the resulting optimized overlap is shown. 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) Nonoptimized

b) Optimized overlap

Fig. 5 Three-mesh example.

Projection for Viscous Grids

The oversetting process gives the user great flexibility in how the surface of a body is subdivided 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 Fig. 6, which shows two overlapping grids on a curved surface. The scale of these grids and the curvature of the surface are exaggerated in Figs. 6 to clarify the problem. The first problem type occurs for a concave surface, as seen in the top half of Fig. 6. 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 the viscous interpolation problem is easily identified by the presence of orphan points near a curved surface. The second form of viscous interpolation errors occurs for a convex surface (bottom half of Fig. 6) 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 farther 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.

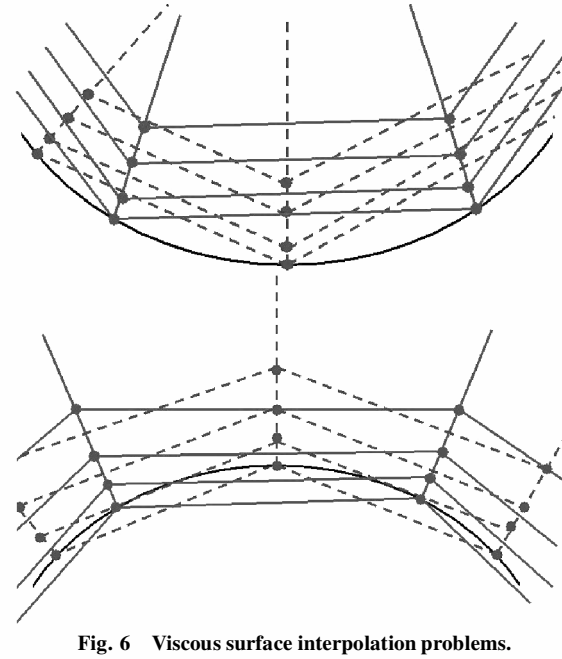


Fig. 6 Viscous surface interpolation problems.

To correct this problem, some existing software was incorporated into the PEGASUS 5 code to perform overset-grid projections. This was based on the PROGRD code within the Chimera Grid Tools package.[16] Within this implementation, the grid coordinates are shifted to account for the distance required to project one surface onto the other, as depicted in Fig. 7. These shifted coordinates are then used only in determining the interpolation indices and coefficients. The original unshifted grid coordinates are used in all other operations in the software. The first step in the projection process is to project the recipient mesh onto the donor mesh; points in

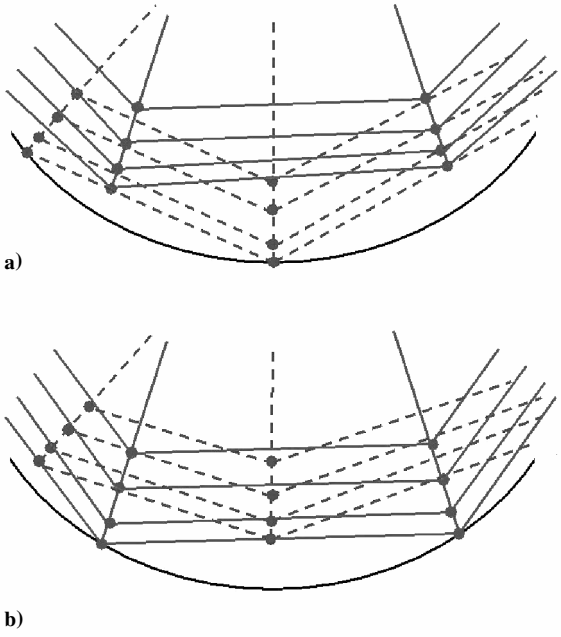


Fig. 7 Mesh projection by mesh pair: a) ——, recipient mesh and – – –, donor mesh and b) – – –, recipient mesh and ——, donor mesh.

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 with the identities of the donor and recipient meshes reversed. This process is repeated for all mesh pairs that have overlapping surfaces.

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 generated earlier, that is, interpolation coefficients between the other grids, is 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.

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 from pairs of meshes. These operations include surface projections between all mesh pairs, building 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, for example, 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. (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 8 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. Notice that most workers become idle for a brief time at the completion of some of the processes, such as the projections, the ADT operations, and the auto hole cutting. This particular problem executed just under 13 times faster than the execution using only one CPU.

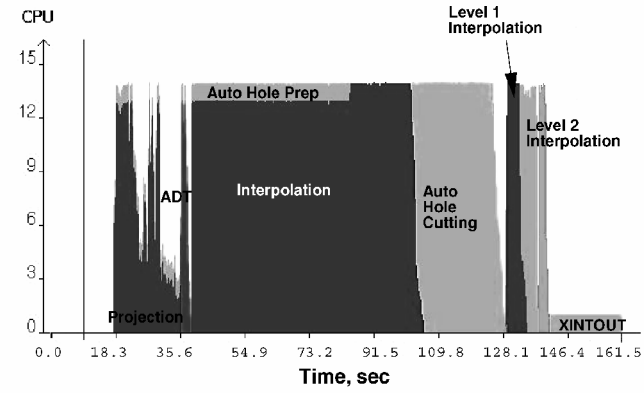


Fig. 8 PEGASUS 5 operations vs time for a 52-mesh example.

For this problem, which consisted of 52 meshes and required 2429 separate operations, 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. 8, is performed in serial by just one processor. This operation is the last step in the code in which all of the final interpolation stencils are gathered and written to the final output file. 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 turnaround by a factor of two.

Examples

Two example PEGASUS 5 test cases are presented. They are a three-element high-lift airfoil 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. Flow solutions for these cases have been computed using the results of both PEGASUS 4 and PEGASUS 5 connectivity files. These flow computations utilized the OVERFLOW[7,8] 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. Several utility programs 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 both of these examples.

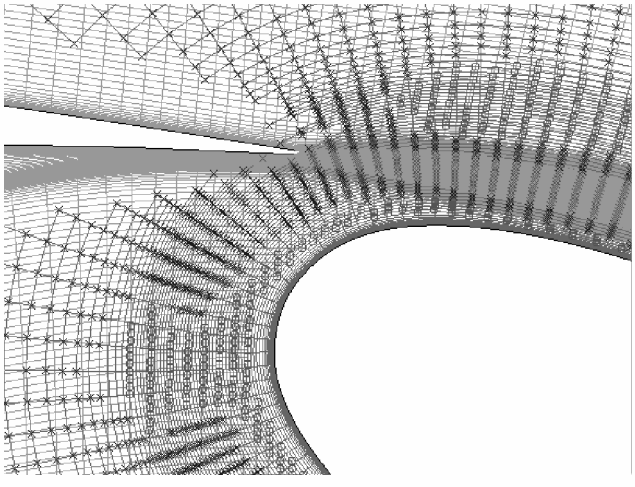
A description of all of the possible input variables recognized by PEGASUS 5 is beyond the scope of this paper. The most commonly used input variable that a user might need to modify to fix problems with the automatic hole cutting operation is the OFFSET input variable. This variable may be specified 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. User modifications to this input variable can fix most of the problems that occur with the automated hole cutting method.

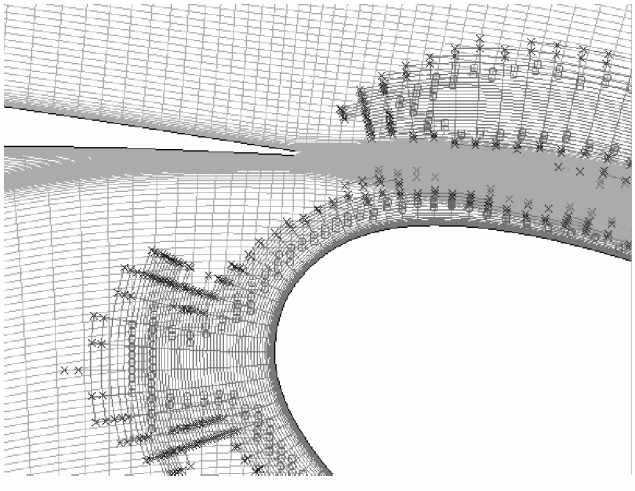
Multi-Element Airfoil

The first test case is a two-dimensional three-element airfoil known as the 30P30N configuration,[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[22] that used the PEGASUS 4 software.

The only modification to the automatically generated PEGASUS 5 input file was to increase the global OFFSET variable to two and to set the OFFSET variable to a value of five for both of the box grids. PEGASUS 5 ran in 65 s on a single SGI R10K 250-MHz CPU. Figure 9 depicts the slat and wing grids at the slat trailing edge and wing leading edge. The symbols indicate the location of fringe points, that are boundary points in a grid that will receive interpolation boundary conditions from its overlapping neighbor. Figure 9a shows all of the fringe points; Fig. 9b shows only fringe points that are one or two points away from an active interior point. Thus, Fig. 9a shows the actual hole cut by PEGASUS 5; Fig. 9b 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. This problem can typically be fixed by increasing the OFFSET input parameter. In addition, as can be seen in Fig. 9b, the effect of the overlap-optimization step 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. 9 Fringe points near the slat trailing edge.

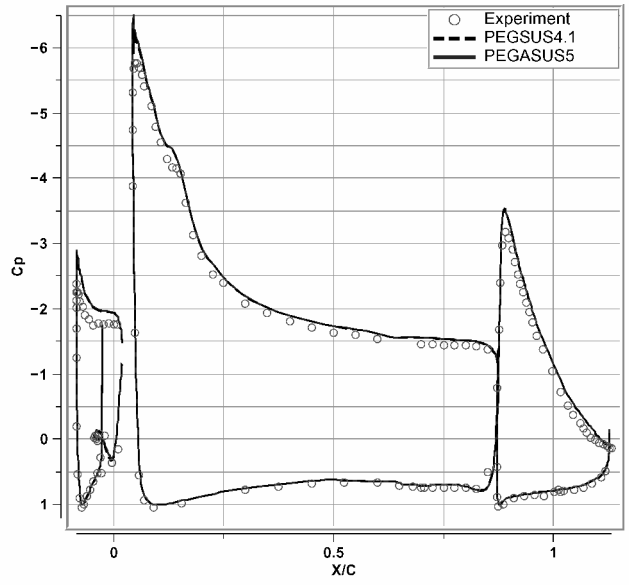


Fig. 10 Pressure coefficient results on three-element airfoil.

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 freestream Mach number was 0.2, the Reynolds number was 9×10^6, and the angle of attack was 8.1 deg. Very similar results were obtained for both grid systems. Figure 10 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.

Boeing 777 Aircraft

The second 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 11 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, outboard flap, and the three largest flap-hinge fairings. The grid system for this geometry was originally developed by Rogers et al.[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 oversetting of the volume grids using the PEGASUS 4 software.

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 min 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

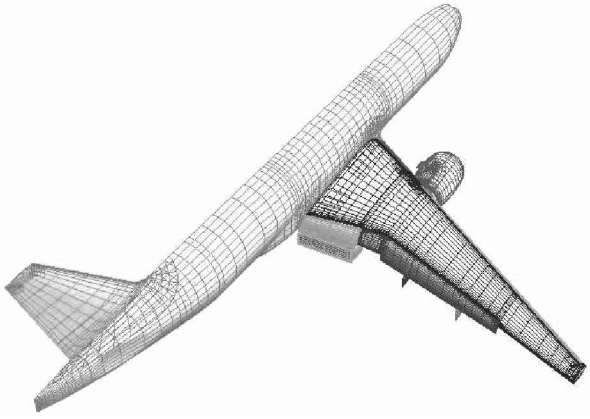


Fig. 11 Surface grids on Boeing 777-200.

magnitude smaller than the fuselage length and the wing semispan. 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 that 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 increased in resolution by 50% in both the longitudinal and vertical directions. After rerunning PEGASUS 5 with these eight automatic hole cutters, the input was further refined by increasing the OFFSET value to one or two for nearly one-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. This circumstance of improper holes being cut can occur at the solid surface of a curved body where overset surface meshes overlap. This results in an ambiguous definition of the solid surface. This creates extremely small gaps in the surface that can cause the line-of-sight algorithm to classify a point improperly as being inside the solid body when it is acutally outside.

After these input-file modifications, a final grid system was obtained that 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. Although these times represent the time required for expert users to perform these tasks, it is significantly easier to become an expert PEGASUS 5 user. Also, the algorithms, methods, and inputs are so different between the two versions that the experience of running this problem in PEGASUS 4 did not help to reduce the time required to run PEGASUS 5 for this problem.

Subsequent runs of PEGASUS 5 for the 777 grid system were performed to test the parallel performance of the code. The parallel speedup for the 777 was very good. The 777 grids were run on 48 SGI Origin processors, which provided a speedup of a factor of 33 over the use of a single processor. By the use of 48 processors, the code was able to process the 777 grids in less than 13 min.

The OVERFLOW code was run using this new PEGASUS 5 grid system 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 5.8×10^6. The lift coefficients for these new runs are plotted

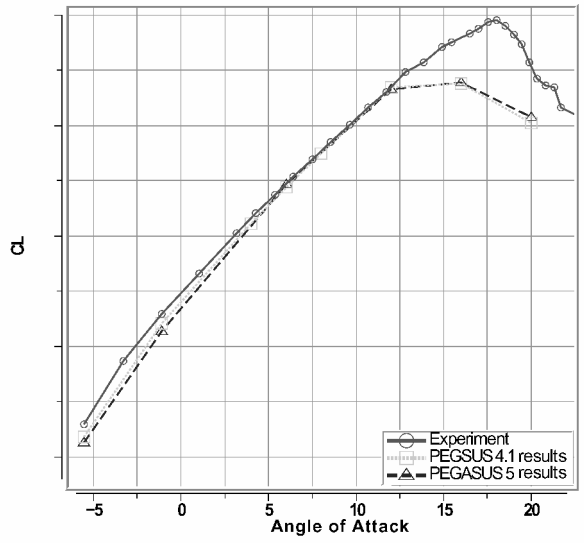


Fig. 12 Lift coefficient vs angle of attack, comparing OVERFLOW results for the PEGASUS 4 and PEGASUS 5 grid systems.

in Fig. 12, together with the earlier computational results and experimental results for this geometry. The actual value of the lift coefficient is not included on the vertical axis labels due to the proprietary nature of these 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, farther 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 deg and lower, and, at typical approach conditions, the computed lift is within 1.5% of the experimental lift.

Performance

Here, a breakdown of the computational cost of each process is given, to quantify the performance of the current algorithms. The average percentage of the primary processes for each of the two test cases is given. The projection process used 10% of the processing time, the interpolation process used approximately 50% of the total time, the hole cutting processes used about 30% of the time, and the overlap optimization used about 5% of the time. The cost of the hole-cutting operation is the second most expensive step in the code. A typical three-dimensional problem will require that approximately 5–15% of the grid points be blanked out in the hole cutting procedure. The current algorithm processed hole points at the rate of 100–200 blanked points per CPU second on an SGI R10K 250-MHz workstation.

It is to be expected that the interpolation process uses more time than any other process because it searches for all possible donor cells for every single grid point. In fact, the interpolation process typically produces an average of two or three donor cells for each grid point. The computational performance of the interpolation scheme is on the order of 10,000 donor cells per CPU second on an SGI R10K 250-MHz workstation. Approximately 5% of the donor grid cells found in the interpolation process are ultimately kept and stored in the final output file. Although this approach may seem excessive or even wasteful, the goal of the current work was to produce a fully automatic code. Indeed, the increased computational cost appears to be well worth the minimal amount of user time and expertise that is required by PEGASUS 5.

The PEGASUS 5 code enables a user to perform overset-grid connectivity effectively for a complex three-dimensional problem in one day of labor time, which represents at least an 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 of

labor time spent changing inputs and rerunning the code to perform such an operation[24] and required significantly greater user expertise than is required by the PEGASUS 5 software. The realization of a production-ready code to perform the overset preprocessing automatically takes a big step toward realizing the 1996 strategic goal of the NASA/Boeing Advanced Subsonic Program of reducing cycle time for complex three-dimensional problems from hundreds of days to five days.[24]

Conclusions

The newest version of PEGASUS, Version 5, has been automated to reduce the number of user inputs and the time required to determine the interconnectivity 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 the projection process improves the interconnectivity solutions that are produced by PEGASUS 5. The new algorithms used in PEGASUS 5 have been found to work very well. The only shortcomings that typically require user intervention are two problems that can occur with the automated hole cutting. The first of these is the failure to cut out hole points that are inside a very thin body, such as the trailing-edge region of a wing. These points are easily fixed by increasing the OFFSET input variable or increasing the resolution of the Cartesian hole map. The second failure is improper holes being cut at the solid surface of a curved body where overset surface meshes result in an ambiguous definition of the solid surface. This failure is less common and can be fixed by specifying a subset of the grid where the code can not cut any holes.

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 speedup for a modest number of processors, depending on the problem size. A speedup of over a factor of 33 was obtained on 48 SGI Origin processors 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 that that required by the PEGASUS 4 code for processing the same volume grids.

Acknowledgments

This work was partially funded by the Advanced Subsonic Technology Program through NASA Contract NAS2-97032 and by the NASA High Performance Computing and Communications Program. The authors acknowledge Robert Meakin of the U.S. Army Aeroflightdynamics Directorate at NASA Ames Research Center and Tom Olsen of Eloret Corp. at NASA Ames Research Center for their helpful comments during the writing of the manuscript.

References

[1]Steger, J. L., Dougherty, F. C., and Benek, J. A., "A Chimera Grid Scheme," *Advances in Grid Generation*, edited by K. N. Ghia and U. Ghia, American Society of Mechanical Engineers, Fairfield, NJ, Vol. 5, 1983, pp. 59–69.

[2]Benek, J. A., Dougherty, F. C., and Buning, P. G., "Chimera: A Grid-Embedding Technique," Arnold Engineering Development Center, AEDC-TR-85-64, Arnold AFB, TN, Dec. 1985.

[3]Meakin, R. L., "Object X-Rays for Cutting Holes in Composite Overset Structured Grids," AIAA Paper 2001-2537, June 2001.

[4]Belk, D. M., and Maple, R. C., "Automated Assembly of Structured Grids for Moving Body Problems," AIAA 95-1680, June 1995.

[5]Wang, Z. J., Parthasarathy, V., and Hariharan, N., "A Fully Automated Chimera Methodology for Multiple Moving Body Problems," AIAA Paper 98-0217, Jan. 1998.

[6]Brown, D. L., Henshaw, W. D., and Quinlan, D. J., "Overture: Object-Oriented Tools for Overset Grid Applications," AIAA Paper 99-3130, June 1999.

[7]Kandula, M., and Buning, P. G., "Implementation of LU-SGS Algorithm and Roe Upwinding Scheme in OVERFLOW Thin-Layer Navier–Stokes Code," AIAA Paper 94-2357, June 1994.

[8]Jespersen, D. C., Pulliam, T. H., and Buning, P. G., "Recent Enhancements to OVERFLOW," AIAA Paper 97-0644, Jan. 1997.

[9]Tramel, R., and Nichols, R., "A Highly Efficient Numerical Method for Overset-Mesh Moving-Body Problems," AIAA Paper 97-2040, June 1997.

[10]Rogers, S. E., Kwak, D., and Kiris, C., "Numerical Solution of the Incompressible Navier–Stokes Equations for Steady-State and Time-Dependent Problems," *AIAA Journal*, Vol. 29, No. 4, 1991, pp. 603–610.

[11]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, 1994, pp. 886–891.

[12]Benek, J. A., Donegan, T. L., and Suhs, N. E., "Extending the Chimera Grid Embedding Scheme with Applications to Viscous Flow," AIAA Paper 87-1126, June 1987.

[13]Dietz, W. E., and Suhs, N. E., "PEGSUS 3.0 Users Manual," Arnold Engineering Development Center, AEDC-TR-89-7, Arnold AFB, TN, Aug. 1989.

[14]Suhs, N. E., and Tramel, R. W., "PEGSUS 4.0 Users Manual," Arnold Engineering Development Center, AEDC-TR-91-8, Arnold AFB, TN, Nov. 1991.

[15]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.

[16]Chan, W., "The OVERGRID Interface for Computational Simulations on Overset Grids," AIAA Paper 2002-3188, June 2002.

[17]Chiu, I. T., and Meakin, R., "On Automating Domain Connectivity for Overset Grids," AIAA Paper 95-0854, Jan. 1995.

[18]Bonet, J., and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," *International Journal of Numerical Methods in Engineering*, Vol. 31, 1991, pp. 1–17.

[19]Meakin, R. L., "Composite Overset Structured Grids," *Handbook of Grid Generation*, edited by J. F. Thompson, B. K. Soni, and N. P. Weatherill, CRC Press, New York, 1999, pp. 11-1–11-20.

[20]Henshaw, W. D., "Ogen: An Overlapping Grid Generator for Overture," Lawrence Livermore National Lab., Research Rept. UCRL-MA-132237, Livermore, CA, 1998.

[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," *Journal of Aircraft*, Vol. 38, No. 6, 2001, pp. 1060–1068.

[24]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, Aug. 2000.

P. Givi
Associate Editor