CHAPTER 25

Application of Domain Decomposition to the Analysis of Complex Aerodynamic Configurations*

W.E. Dietz
J.L. Jacocks
J.H. Fox

ABSTRACT. An application of domain decomposition to the analysis of aerodynamic configurations is presented. An aerodynamic configuration is discussed which consists of approximately 40 interacting meshes, modeling an aircraft fuselage, an attached wing, pylons attached to the wing and fuselage, stores attached to the pylons, and inlet walls. The method used to analyze this configuration employs two computer codes. The first calculates interpolation information among interacting meshes; the second solves the Euler equations for the entire configuration, using the interpolation information generated by the first code. Issues related to mesh interactions, modeling complex geometries, and computational accuracy are discussed.

1.0 INTRODUCTION. The Arnold Engineering Development Center (AEDC) performs wind tunnel testing on a wide range of military aircraft. The aircraft configurations, an example of which is depicted in Fig. 1, often include stores suspended from pylons attached to both the wings and fuselage. In addition, the test configurations often include active engine inlets.

Computational fluid dynamics (CFD) calculations in support of aircraft ground testing are an important aspect of the AEDC mission. Aerodynamic problems related to wind tunnel testing present two important challenges to CFD: 1) the requirement to produce timely solutions in response to testing requirements, and 2) the solution of problems related to complex three-dimensional configurations in a

*The research reported herein was performed by the Arnold Engineering Development Center (AEDC), Air Force Systems Command. Work and analysis for this research were done by personnel of Calspan Corporation/AEDC Operations, operating contractor for the AEDC aerospace flight dynamics facilities. Further reproduction is authorized to satisfy needs of the U. S Government.

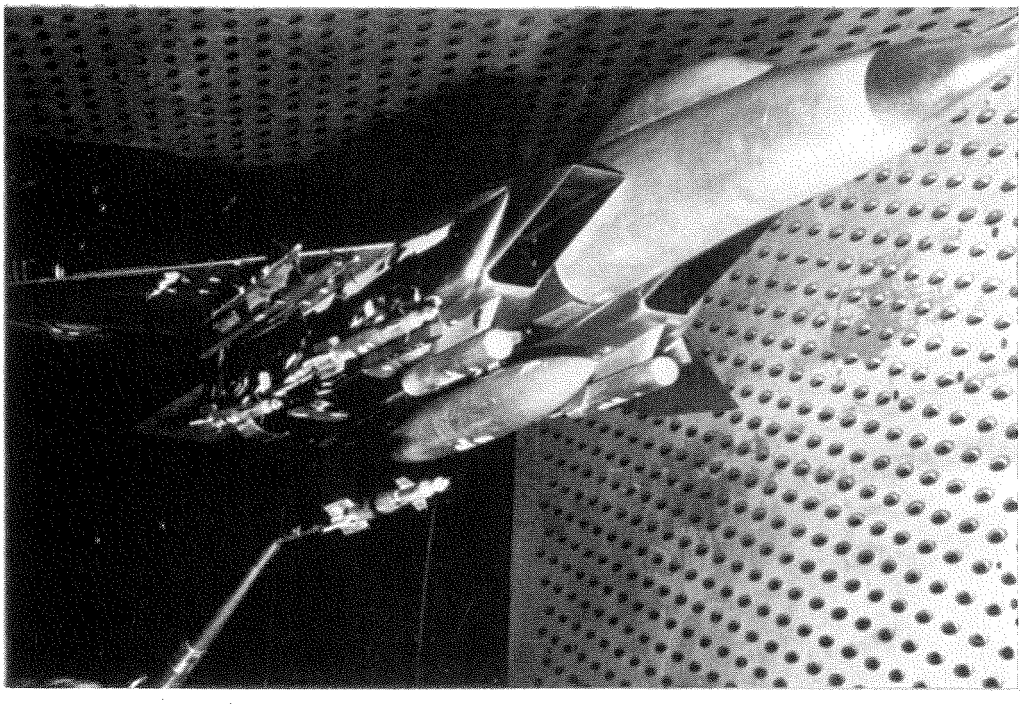


Fig. 1. Complex aerodynamic configuration.

transonic flow regime. The main obstacle to overcoming these challenges is the difficulty of generating computational meshes. Mesh generation is time-consuming, even for relatively simple three-dimensional configurations. Generating a single mesh for the configurations often encountered in tunnel testing may be impractical or topologically impossible. In addition, a single mesh which is fine enough to resolve the desired aerodynamic features of a complex flow field may be too large for available computer memory.

To address these problems, Benek, et al. (Refs. 1-4) have developed a method of domain decomposition called chimera, which allows a system of relatively simple grids, each describing a component of a complex aerodynamic configuration, to be combined into a composite grid to yield solutions to complex flow fields. The chimera scheme is general in that it allows solid surfaces in a mesh to be embedded within the computational domains of other meshes. In addition, overlapping mesh outer boundaries are allowed. The chimera scheme is presently used with both Euler and thin-layer Navier-Stokes flow solvers to obtain flow field calculations about highly complex three-dimensional aircraft configurations, and has been applied to transonic store separation problems (Ref. 5), cavity flows (Ref. 6), and transonic tunnel wall interference calculations (Ref. 7).

1.1 GENERAL CONCEPT. The concept behind the chimera scheme is illustrated in Fig. 2, which depicts two independently generated meshes modeling 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 elements of the airfoil mesh. However, a reverse process must occur as well; the flap mesh must communicate flow field information to the airfoil mesh.

Since the airfoil mesh has no boundary through which flow field information can be obtained from the flap mesh, an artificial boundary must be defined within the airfoil mesh. Mesh points on the artificial boundary which are defined within the airfoil mesh and which are fully contained within the computational region of the flap mesh can be updated by interpolation from the appropriate mesh elements of the flap mesh. Generally, any mesh can receive information from other meshes through outer boundary and artificial boundary points.

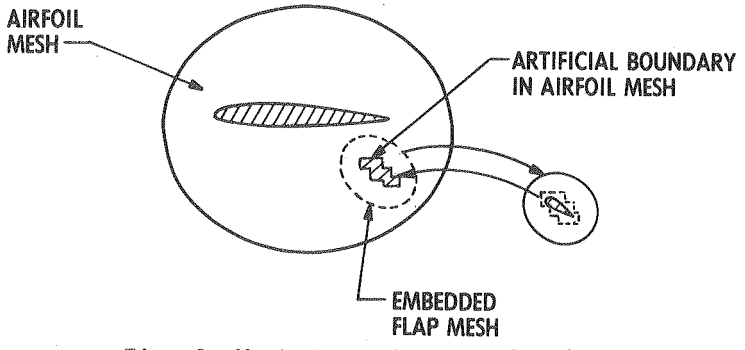


Fig. 2. Mesh-to-mesh communication.

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

1.2 FUNCTIONAL DESCRIPTION OF THE CHIMERA SCHEME. Application of the chimera scheme requires two main steps: 1) a description of how each mesh is to communicate flow-field information to other meshes, and 2) execution of a flow solver which utilizes the communication information generated in the previous step. Presently, two computer programs, PEGSUS and XMER3D, perform steps 1 and 2, respectively. The processes accomplished by PEGSUS include establishing which boundary points in a mesh can be updated by interpolated flow variables from other meshes, and calculating the required interpolation coefficients for each mesh element sending information to a boundary point. The relation of PEGSUS to the chimera domain decomposition scheme is depicted schematically in Fig. 4. The individual meshes and user-defined mesh connection data are input into PEGSUS, which produces 1) a composite mesh, i.e., a single file consisting of the concatenation of all meshes in the multiple mesh configuration, and 2) an interpolation file, which is a table associating all boundary points in the composite mesh with mesh elements which supply the boundary points with interpolated flow-field parameters. The composite mesh and interpolation file are input into the XMER3D flow solver, which calculates the flow field in the composite mesh configuration.

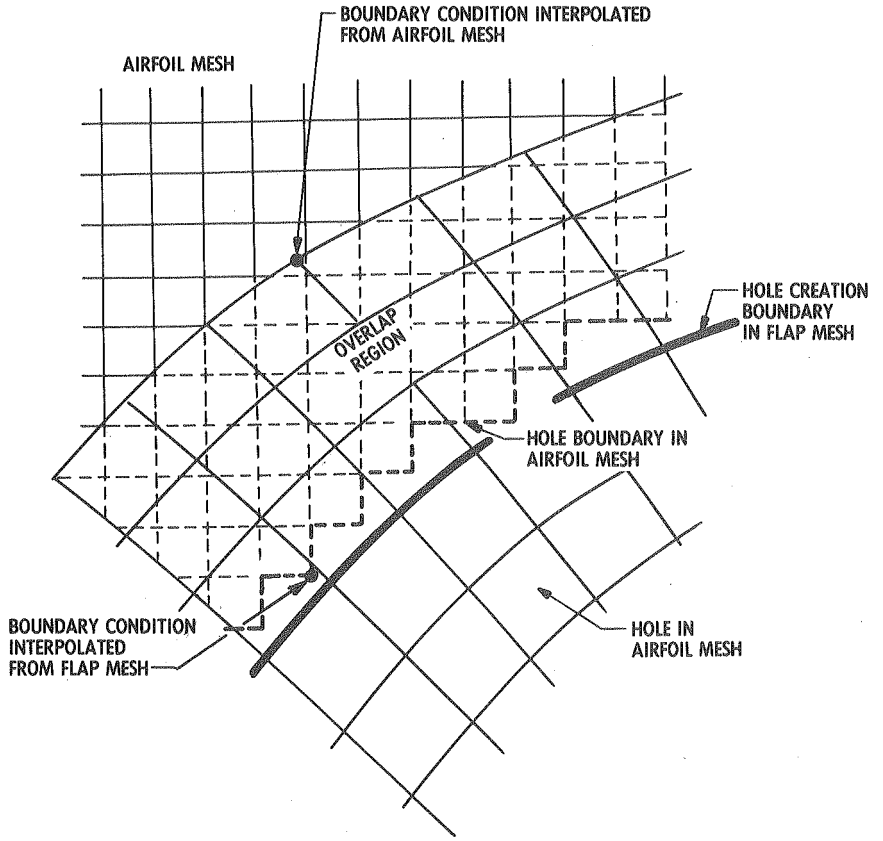


Fig. 3. Overlap region between grids.

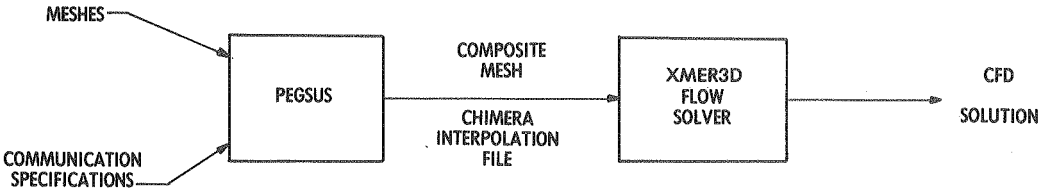


Fig. 4. The Chimera scheme.

The input and output to PEGSUS is depicted in Fig. 5. PEGSUS requires as input the individually generated meshes which define the configuration, and a description of how the meshes interact with one another. The output of PEGSUS, in addition to the composite mesh and interpolation file, consists of a summary of connection information, diagnostic maps which graphically depict mesh connections, and an information file which lists every boundary point and its corresponding interpolation element in the composite mesh.

2.0 MESH, BOUNDARY, AND SURFACE INTERACTIONS. Mesh interactions in PEGSUS are defined by three types of relationships:

- mesh-mesh
- mesh-boundary
- surface-boundary

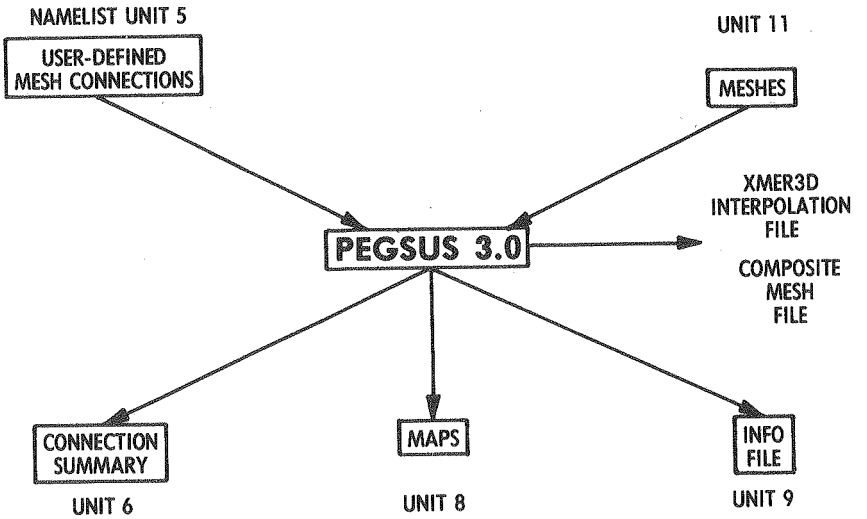


Fig. 5. PEGSUS input and output files.

Figure 6 depicts the type of information which is supplied in the input to define mesh-mesh connections. Each mesh in the composite mesh receives information from other meshes in one of two ways, either through its outer boundary or through hole boundaries which have been created by hole creation boundaries in other meshes. For convenience, hole boundary and outer boundary linkages are referred to as HBLINKs and OBLINKs, respectively. These linkages are priority lists which indicate the order in which other meshes will be searched for interpolation elements. Fig. 6 indicates that Mesh 1 will search Mesh 2, Mesh 3, and Mesh N, in that order, for interpolation elements which can provide information to the hole boundary points of Mesh 1. Mesh 2 and Mesh 5 will be searched in a similar manner for interpolation elements which can provide information to the outer boundary of Mesh 1, while Mesh 4 will not be searched at all. Each mesh in the composite mesh will be related to the other meshes by similar linkages. PEGSUS allows up to 30 outer boundary and 30 hole boundary linkages to be defined for each mesh; more linkages can be added, if necessary, with minimal programming effort.

Figure 7 depicts the relationship between meshes and boundaries. Boundaries are defined as collections of level surfaces within meshes. OB 1 is the outer boundary of Mesh 1, while HB 1 and HB 2 are hole creation boundaries defined within Mesh 1. Since OB 1 is defined in terms of surfaces within Mesh 1, there exists a specific relationship between OB 1 and Mesh 1. This relationship is designated ISPARTOF, i.e., OB1 "ISPARTOF" Mesh 1. The same ISPARTOF relation holds between the hole creation boundaries and Mesh 1. Hole creation boundaries also have relationships to other meshes, i.e., they cause holes in other meshes. This relationship is denoted as "MHOLEIN" (Makes a HOLE IN) in the present discussion. According to Fig. 7, HB 2 makes holes in both Mesh 2 and Mesh 3, while HB 1 makes a hole in Mesh N. Meshes 4 and 5 are not affected by hole creation boundaries defined within Mesh 1. PEGSUS allows each mesh to create holes in up to 30 other meshes.

Finally, Fig. 8 illustrates the relationship between surfaces and boundaries. Boundaries are comprised of collections of surfaces, and therefore surfaces have ISPARTOF relationships to boundaries in much

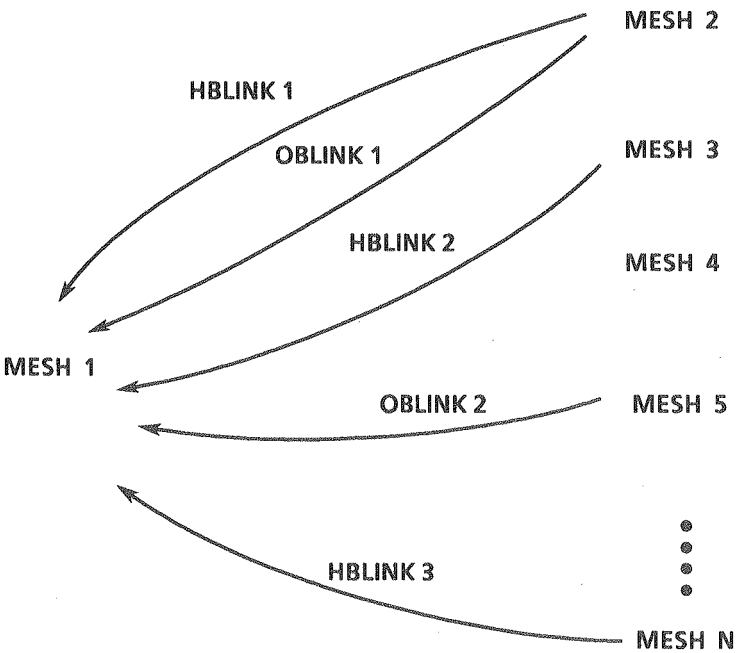


Fig. 6. Mesh-mesh connections.

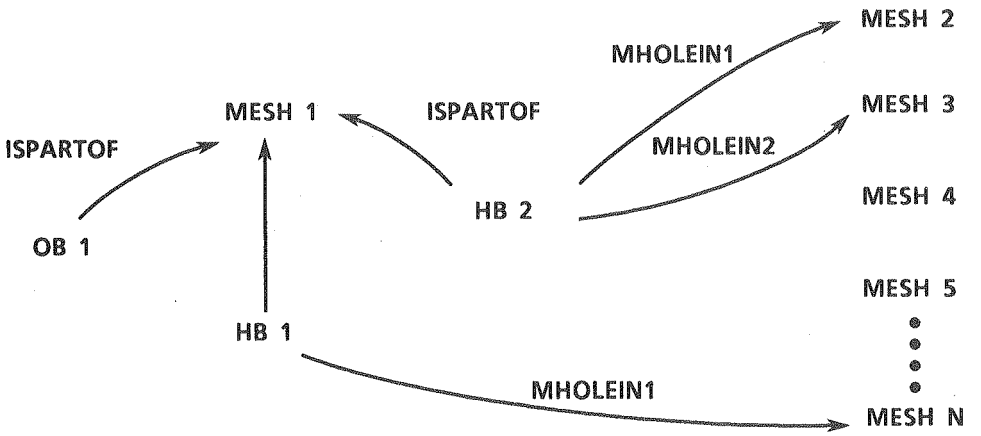


Fig. 7. Mesh-boundary relations.

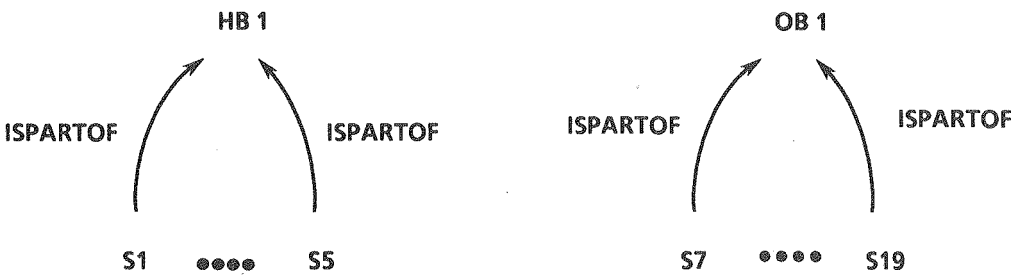


Fig. 8. Surface-boundary relations.

the same way as boundaries do to meshes. There is no limit in PEGSUS to the number of surfaces which may comprise a boundary.

The relationships between meshes, boundaries, and surfaces for the airfoil/flap configuration depicted in Figs. 1 and 2 is illu-

strated in Fig. 9. The flap outer boundary receives interpolated information from the airfoil mesh; therefore, an OBLINK is defined from the airfoil mesh to the flap mesh. The HBLINK to the airfoil mesh from the flap mesh indicates that the airfoil hole boundary receives interpolated information from the flap mesh. Two boundaries are specified within the flap mesh: a hole creation boundary, and the flap mesh outer boundary. The hole creation boundary makes a hole in the airfoil mesh; accordingly, an MHOLEIN link exists between the hole creation boundary and the airfoil mesh. Assuming an O-mesh topology for the flap mesh with computational coordinates of J, K, and L, with the flap surface occurring at L = 1, a single surface suffices to define the hole creation boundary (e.g., L = 5) and the flap outer boundary (maximum L, or LMAX).

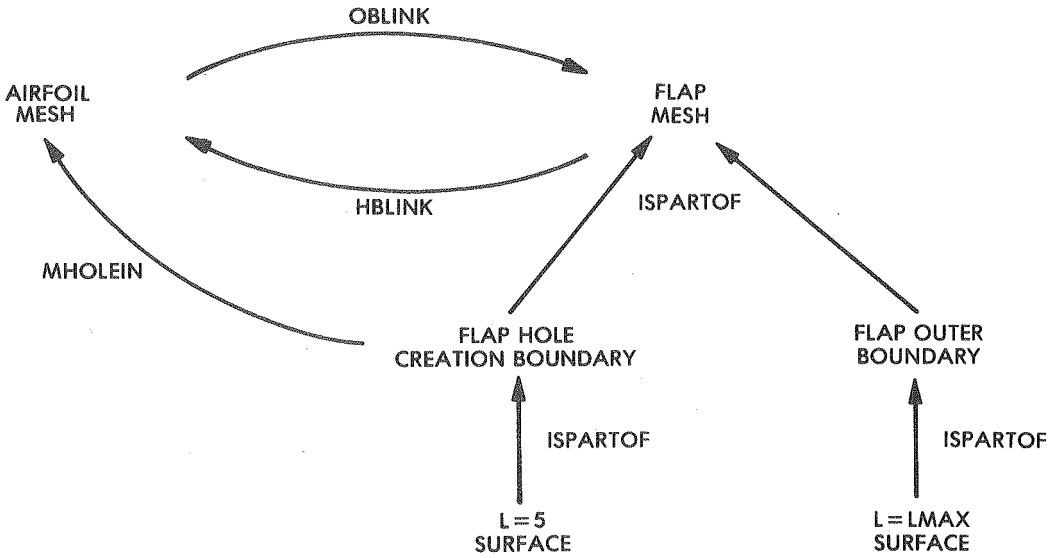


Fig. 9. Mesh, boundary, and surface relationships for airfoil/flap configuration.

The relationships illustrated in Fig. 9 will not change if the angle of attack of the flap is varied, as long as the airfoil surface does not impinge on the outer boundary of the flap mesh. (In that case, the airfoil should make a hole in the flap mesh.) In addition, if the topology of the flap mesh is altered (e.g., to a C-mesh topology), the only changes required in the relationships depicted in Fig. 9 will be the surface definitions which comprise the hole creation and outer boundaries of the flap mesh. The flexibility of the surface and boundary descriptions allow relationships to be described between meshes, boundaries, and surfaces regardless of mesh topology.

3.0 PEGSUS. PEGSUS performs four main processes: hole location, generation of hole boundary points, hole boundary point interpolation, and outer boundary point interpolation. The four functions are performed in the order indicated, and only after all previous functions have been completed on the entire composite mesh. This modularity allows PEGSUS to be executed incrementally, i.e., the functions of hole and outer boundary point interpolation can be performed by "restarting" from output generated from previous functions. As a result, errors in input or mesh configuration can be

identified and corrected at each stage before continuing to the next function.

3.1 HOLE LOCATION. Identifying hole points requires determining whether a mesh point is inside or outside a hole creation boundary defined in another mesh. A mesh point is considered to be inside a hole creation boundary of another mesh if it is inside all surfaces which define the boundary. Figure 10 illustrates the method used to determine whether a point is inside or outside a surface. A mesh point is considered to be inside a surface if the dot product between the vector from the closest point on the surface to the field point (shown as $\vec{R}$), and the normal vector on the surface at the closest point ($\vec{N}$), is negative or zero. (Note that the normal vector $\vec{N}$ must be defined as being directed outward from the hole region). If the dot product is positive, the mesh point is considered to be outside the surface.

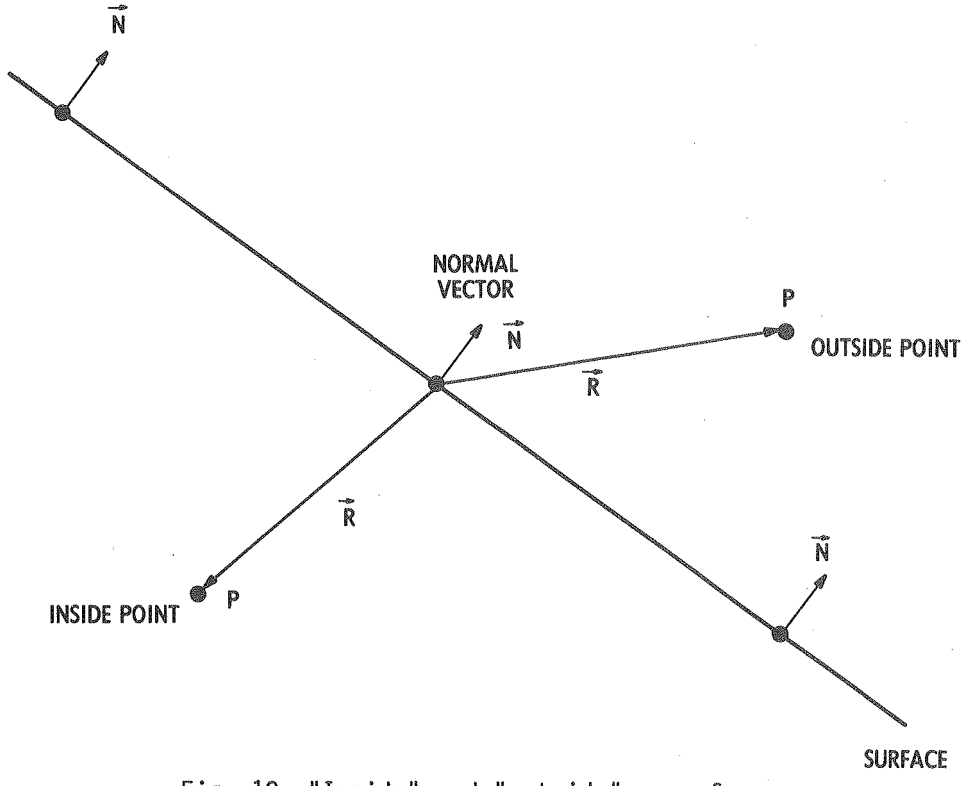


Fig. 10. "Inside" and "outside" a surface.

Figure 11 illustrates the hole location process for a hole creation boundary consisting of the three surfaces S1, S2, and S3. PEGSUS first puts the indices of all points of the mesh in which the hole is to be created into a list. Then, all points which are outside the first surface are eliminated from the index list. The process is then repeated; all points which are contained in the shortened list and which are outside the second surface are eliminated, shortening the index list further. When the process is repeated for the third surface, the points remaining in the list are the hole points of the mesh in which the hole is created.

This approach to hole generation requires that the hole creation boundary be completely closed. For instance, it is clear that leaving

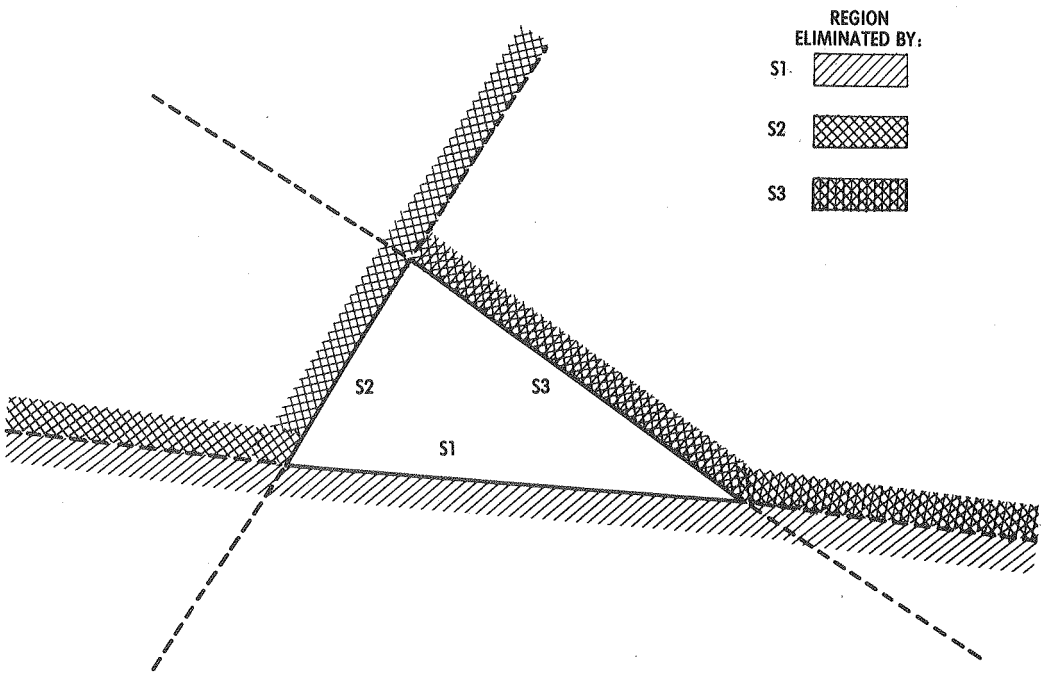
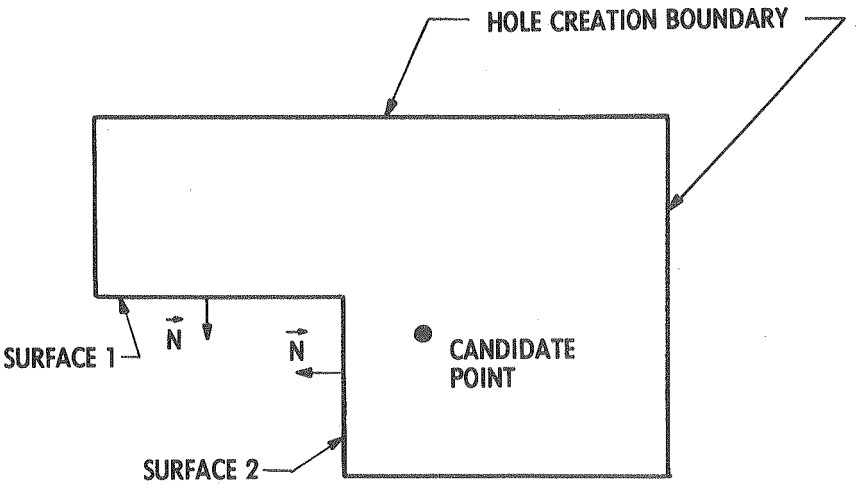


Fig. 11. Hole location.

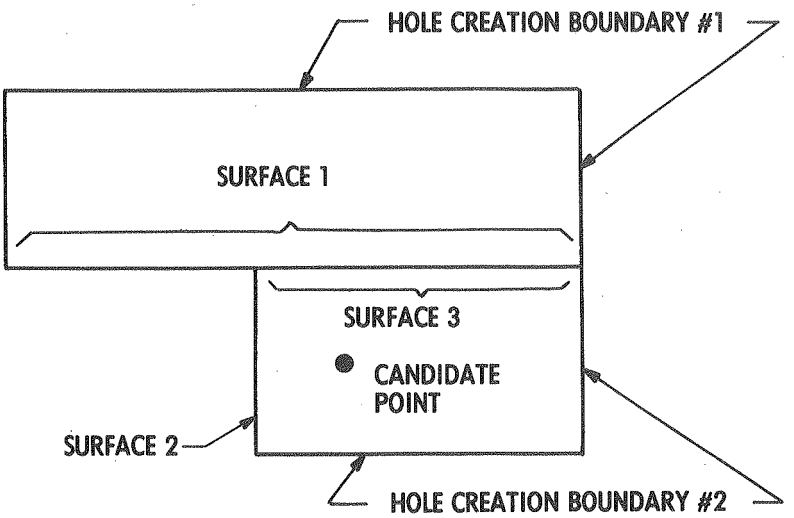
off any one of the boundaries in Fig. 11 will result in many points being incorrectly designated as hole points. The requirement of closed hole creation boundaries often cannot be satisfied adequately with certain types of configurations (e.g., some wing meshes). However, techniques have been developed to satisfy the closed-boundary requirement for these cases, and will be discussed in Sec. 5.1.

The method used in PEGSUS to find hole points has another important restriction, which is illustrated in Fig. 12. In Fig. 12a, a hole creation boundary is defined. The candidate boundary point will be considered to be outside surface 1, and therefore will be considered to be outside the entire hole boundary. As a result, the point will not be identified as a hole point. Generally, hole creation boundaries (as viewed from the outside) must be convex to obtain correct results. However, PEGSUS allows multiple holes to be defined within a mesh. Any hole creation boundary which contains concavities can be divided into one or more entirely convex regions, each of which can create holes in one or more other meshes. If the hole creation boundary is redefined as two separate convex boundaries, as in Fig. 12b, the candidate point will be inside one of the hole creation boundaries, and will therefore be correctly identified as a hole point. The candidate points within the two boundaries will all be labeled as hole points and will therefore effectively merge into a single hole.

3.2 HOLE BOUNDARY (FRINGE) POINT GENERATION. Once all holes in all meshes are correctly located, mesh points surrounding holes (i.e., hole boundary points, also called fringe points in chimera terminology) must be identified. The identification of hole boundary points (in two dimensions) is illustrated in Fig. 13. Hole boundary points are, by definition, always adjacent to hole points. Each hole



a. Hole creation boundary with concavity



b. Removal of concavity

Fig. 12. Restrictions on hole creation boundaries.

point has six adjacent points in three dimensions (in PEGSUS nomenclature, a hole point and its adjacent points comprise a seven-point stencil). Any adjacent point which is not itself a hole point is identified as a candidate hole boundary point. The final result is the generation of a list of candidate hole boundary points which are to receive interpolated flow-field information from other meshes.

3.3 HOLE BOUNDARY INTERPOLATION. Hole boundary interpolation is performed after all hole points and candidate hole boundary (i.e., fringe) points in the composite mesh have been identified. At this point PEGSUS searches other meshes in the composite mesh for mesh elements which may be used to interpolate flow field information to the hole boundary points.

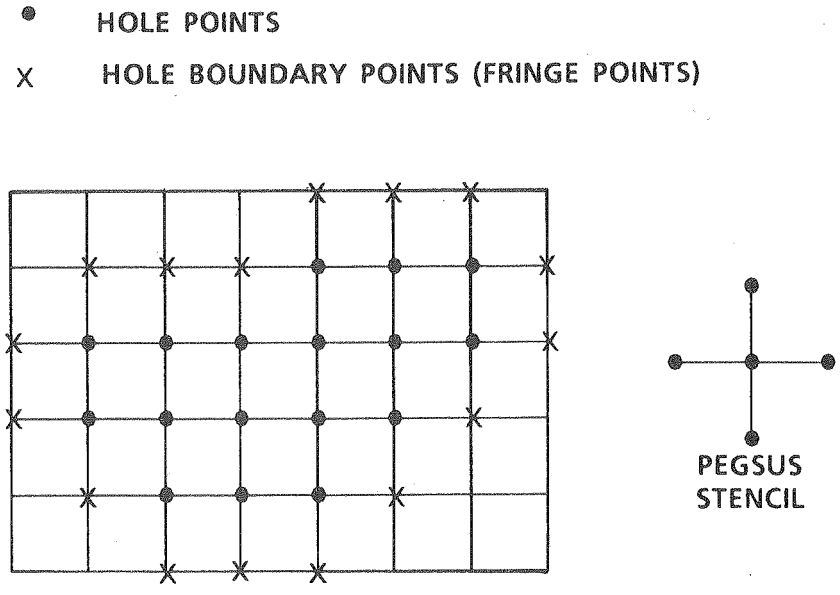


Fig. 13. Hole boundary generation (2-D).

3.3.1 HOLE BOUNDARY PRIORITY LIST. The HBLINK definitions for each mesh (Sec. 2.0) comprise a priority list containing the names of other meshes which are to be searched for interpolation elements which can provide information to hole boundary points. The meshes in the priority list are searched in the order in which they appear in the input. If an interpolation element is found, the interpolation information for the point is stored in a set of arrays. If the hole boundary points cannot be interpolated by mesh elements in the first mesh in the priority list, the second mesh is searched for interpolation elements. The process continues until either interpolation elements for all hole boundary points have been found, or the mesh priority list is exhausted. Any hole boundary points which remain after the mesh priority list is exhausted are termed "orphans." Orphaned hole boundary points will be treated as hole points and hence will not be updated by the flow solver.

Since hole location and hole boundary point interpolation occur as two separate functions, a mesh which causes a hole in another mesh does not necessarily have to be the mesh which provides interpolated information to the resultant hole boundary points. This allows meshes to be used for no other function than to generate holes in other meshes; this feature is used extensively in certain instances, such as the modeling of a fuselage and attached wing (Sec. 5.1).

3.3.2 VALID AND INVALID INTERPOLATIONS. Finding a mesh element which can supply interpolated information to a hole boundary point is a necessary but not sufficient condition for an interpolation to be considered satisfactory. For instance, a mesh element which is to supply interpolated information to a boundary point may itself be comprised of mesh boundary points which are updated through interpolation. As a result, most of the information used to update the original boundary point will come from interpolation, rather than the solution of the flow field. This close coupling of boundary points often results in poor global convergence. The strategy used in PEGSUS is to

simply not allow boundary points to be updated by mesh elements which are themselves comprised of boundary points.

Figure 14 illustrates the difference between valid and invalid interpolations, as defined within PEGSUS. In Fig. 14, the outer boundary of a mesh (dotted lines) is to receive interpolated information from another mesh (solid lines). (We will refer to the first mesh as the "recipient" mesh, and the second mesh as the "donor" of interpolated information). An interpolation is valid if an interpolation element can be found in the donor mesh for which none of the corner points of the element are hole or boundary points, or if a boundary point is coincident with a corner of an interpolation element in the donor mesh, and the corner point itself is not a hole or boundary point. The hole boundary point in Fig. 14 comprises a corner of two mesh elements which contain points A and B; as a result, points A and B cannot receive interpolated information from the donor mesh. Point C is coincident with a corner point of the donor mesh. Since the corner point is not itself a boundary or hole point, point C can receive information from the donor mesh. None of the corner points of the mesh element containing point D is a boundary or hole point; therefore, point D will be permitted to receive interpolated information from the donor mesh. In Fig. 14, points A and B will be orphans unless the recipient mesh has an outer boundary link to another mesh which can provide valid interpolation elements.

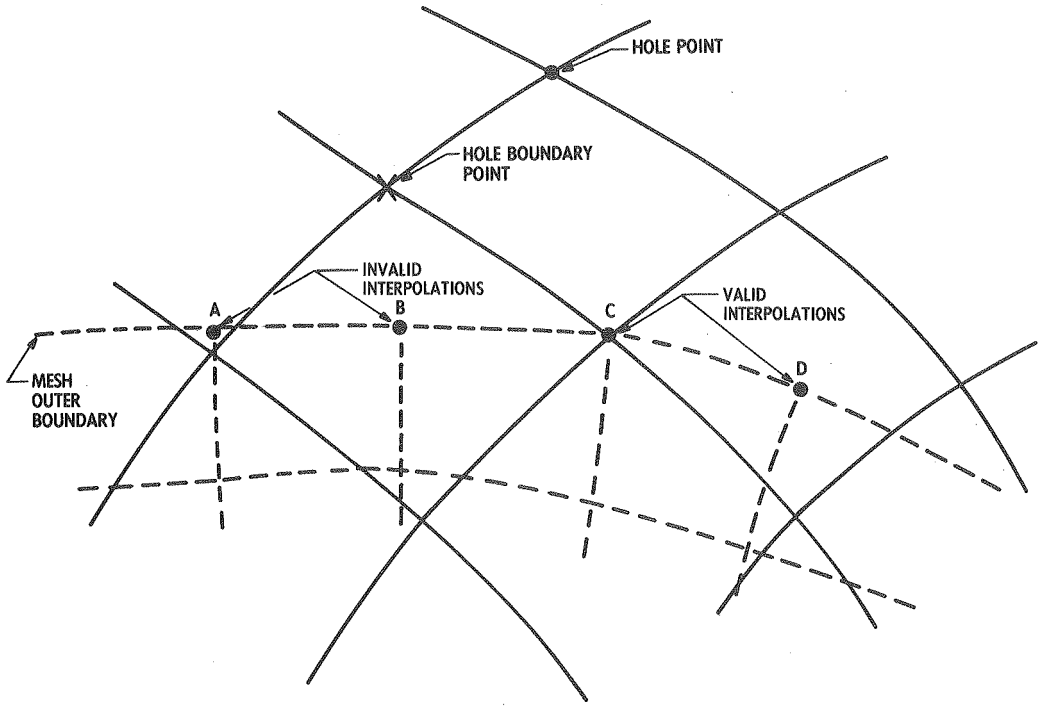


Fig. 14. Valid and invalid interpolations.

3.3.3 SEARCHING FOR INTERPOLATION ELEMENTS. Identifying mesh elements which can be used for supplying interpolated information to boundary points requires some sort of searching procedure. An exhaustive search of mesh elements will always find a permissible mesh element if one exists; however, exhaustive searches are computationally

intensive. PEGSUS uses a heuristic search mechanism which can find an interpolation element in relatively few steps.

The interpolation algorithm used in PEGSUS is a trilinear interpolation scheme, and is fully documented in Ref. 3. The trilinear interpolation algorithm yields interpolation coefficients for a point either inside or outside the element (the latter case is actually an extrapolation). The interpolation coefficients will all be between 0.0 and 1.0 if the point is inside the element; a point outside the element will cause at least one coefficient to be greater than 1.0. The interpolation element in effect defines a local coordinate system, while the interpolation coefficients define the location of the point in the local system. In PEGSUS, the interpolation coefficients are used to determine the direction (in computational coordinates) in which the search should proceed in order to eventually find an element which surrounds the point. For instance, a given boundary point and a mesh element with its lowest-indexed corner point at J, K, and L may yield interpolation coefficients of 5.2, 7.8, and 2.3. PEGSUS will then move to the mesh element corresponding to the point J + 5, K + 7, and L + 2. The new mesh element will usually be closer to the boundary point. The procedure is repeated until all interpolation coefficients are between 0.0 and 1.0. Indices of candidate mesh elements are allowed to jump as much as a third of the distance (in computational coordinates) across a mesh. In general, the search will succeed in three or four steps, even if the initial starting point is far from the final interpolation mesh element. Although this searching mechanism can in principle fail in certain cases (e.g., highly curved or distorted meshes), in practice the searching procedure is highly reliable.

3.4 OUTER BOUNDARY INTERPOLATION. The interpolation of outer boundary points is accomplished in almost exactly the same way as hole boundary interpolation, except that outer boundary points comprise the set of candidate boundary points, and a different priority list (i.e., the OBLINK list, see Sec. 2.0) from the one used for hole boundary processing is used. The criteria for permissible interpolations and the searching algorithm are identical to those used for hole boundary processing. In PEGSUS, only outer boundary surfaces which are to be interpolated are defined in the input. As a result, free-stream outer boundaries do not have to be specified.

Unlike orphaned hole boundary points, orphaned outer boundary points are not necessarily undesirable. For instance, two meshes may have their outer boundaries only partially embedded within each other; if the entire outer boundary of each mesh is designated as a surface to be interpolated, the points on the boundaries outside of any mesh will be designated as orphans. Orphaned outer boundary points are not blanked out, and can therefore have boundary conditions (usually free-stream) imposed by the XMER3D flow solver.

3.5 GENERATION OF INTERPOLATION TABLE. Once the interpolation coefficients are generated for the boundary points in the composite mesh, the interpolation table is generated. The structure of the interpolation table is illustrated in Fig. 15. The interpolation table consists of a list of every boundary point in each mesh which receives interpolated information from other meshes, a list of every mesh element in each mesh which provides interpolated information to boundary points, and pointers from boundary points to the mesh elements which

provide the boundary points with interpolated flow-field data. In addition, PEGSUS outputs a list of all points in the composite mesh that are hole and boundary (i.e., "blanked") points. At this point, processing by PEGSUS is complete; the interpolation table can now be used by XMER3D to update boundary points during flow-field calculations.

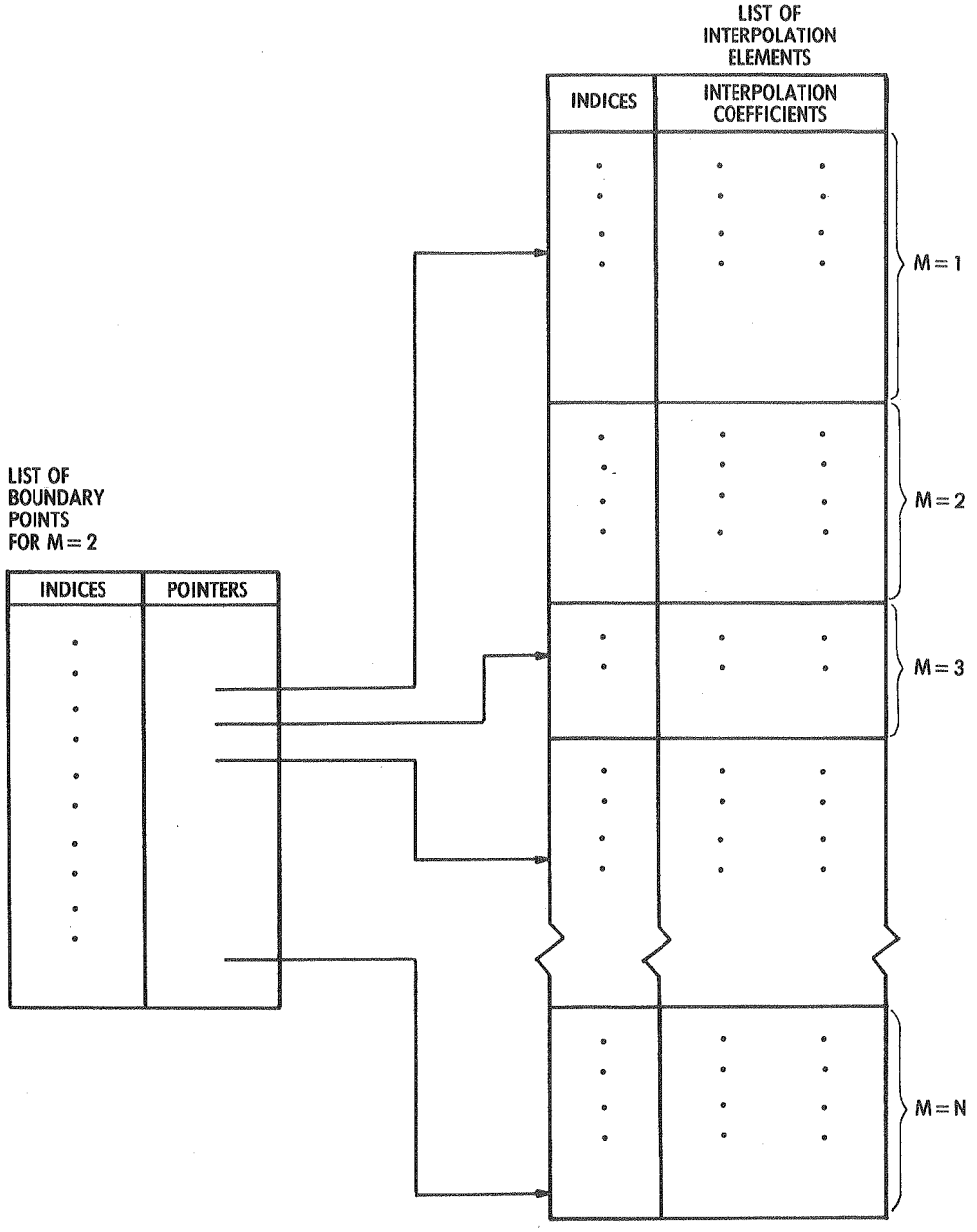


Fig. 15. Data structure used in XMER3D.

4.0 XMER3D. The flow solver code, XMER3D, is a derivative of the AIR3D code developed by Pulliam and Steger (Ref. 8). XMER3D solves the 3-D Euler or Navier-Stokes equations using the implicit approximate factorization algorithm originally developed by Beam and Warming (Ref.

9). Boundary condition imposition is explicit, and is therefore ideally suited for applications using boundary conditions which are interpolated from other meshes.

XMER3D differs from AIR3D in several aspects. AIR3D was designed to work only with single meshes. XMER3D is designed to eliminate from the implicit solution process all points in a mesh that have been identified as hole points or boundary points, and process multiple mesh configurations, using interpolation files generated by PEGSUS. In addition, XMER3D has been extensively vectorized for use on Cray-class machines.

The exclusion of hole points from the solution process is accomplished in a straightforward manner. The implicit algorithm solves a system of equations which may be represented as

$$A\delta = F,$$

where A is a coefficient matrix, δ is the vector of unknowns occurring along a computational coordinate, and F is the known vector. The values of δ correspond to corrections of the latest approximation Φ of the flow variables. Therefore, δ is added to the latest value of Φ to obtain updated values of Φ:

$$\Phi^{n+1} = \Phi^{n} + \delta$$

However, in the case where some components of δ occur at hole or boundary points, it is required that the implicit algorithm calculate correction values equal to zero. For instance, if δ is a vector of seven elements, then the implicit system has the form

$$\begin{pmatrix} a_{11} & a_{12} & 0 & 0 & 0 & & \\ a_{21} & a_{22} & a_{23} & 0 & 0 & & \\ 0 & a_{32} & a_{33} & a_{34} & 0 & 0 & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} & 0 & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} & a_{56} & 0 \\ & & 0 & 0 & a_{65} & a_{66} & a_{67} \\ & & 0 & 0 & 0 & a_{76} & a_{77} \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \delta_5 \\ \delta_6 \\ \delta_7 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{pmatrix}$$

If, for example, the third, fourth, and fifth elements of δ occur at hole or boundary points, corresponding rows of the coefficient matrix A are modified. The off-diagonal elements of relevant rows of the matrix A are set equal to zero, the diagonal elements are set equal to one, and the corresponding elements of F are set equal to zero. The system then takes the form

$$\left[\begin{array}{cc:ccc:cc} a_{11} & a_{12} & 0 & 0 & 0 & & \\ & & & & & 0 & \\ a_{21} & a_{22} & a_{23} & 0 & 0 & & \\ \hdashline & & 1 & 0 & 0 & & \\ & 0 & 0 & 1 & 0 & 0 & \\ & & 0 & 0 & 1 & & \\ \hdashline & & 0 & 0 & a_{65} & a_{66} & a_{67} \\ & 0 & & & & & \\ & & 0 & 0 & 0 & a_{76} & a_{77} \end{array}\right] \left\{\begin{array}{c} \delta_1 \\ \delta_2 \\ \hdashline \delta_3 \\ \delta_4 \\ \delta_5 \\ \hdashline \delta_6 \\ \delta_7 \end{array}\right\} = \left\{\begin{array}{c} F_1 \\ F_2 \\ \hdashline 0 \\ 0 \\ 0 \\ \hdashline F_6 \\ F_7 \end{array}\right\}$$

In this system, the values of the corrections at hole point locations will be zero, as required. In addition, the solutions of δ_1 and δ_2 are uncoupled from the solutions of δ_6 and δ_7. Φ_3 and Φ_5 are values of the flow variables at hole boundary points and are therefore determined by interpolation from other meshes. The corresponding corrections are zero; therefore the interpolated values of Φ are preserved. The modifications of the coefficient matrix are accomplished by XMER3D during execution, using the list of marked or "blanked" points generated by PEGSUS.

PEGSUS also blanks outer boundary points which are to receive interpolated information from other meshes. The same process which excludes hole and hole boundary points from being updated by the flow solver is therefore applied in an identical manner to outer boundary points.

The method by which XMER3D processes multiple meshes is depicted in Fig. 16. The flow solver cycles through every mesh during each time step. During a cycle, boundary conditions are updated and the solution is advanced by two time steps for each mesh. Meshes that are processed later in the cycle will receive updated boundary conditions from meshes which were processed earlier in the cycle. Changes in flow variables in the composite mesh are calculated during each cycle; when the changes fall below a preset tolerance, the composite solution is assumed to be converged to a steady-state solution.

5.0 APPLICATION OF CHIMERA TO A COMPLEX CONFIGURATION. The major impetus behind the development of the chimera scheme was the requirement to perform CFD calculations on complex three-dimensional configurations. Although chimera supplies the basic tools to model general geometries, various aspects of complex configurations must be treated carefully to obtain good results. In the course of applying the chimera scheme, specific methods have been developed to model wing/body junctures, inlets, pylons, and stores.

5.1 MODELING THE WING AND FUSELAGE. A common application of the chimera scheme is the modeling of an aircraft fuselage and wing. This is usually accomplished by generating separate meshes about the fuselage and wing and requiring the meshes to communicate appropriately. A typical wing/body juncture is depicted in Fig. 17.

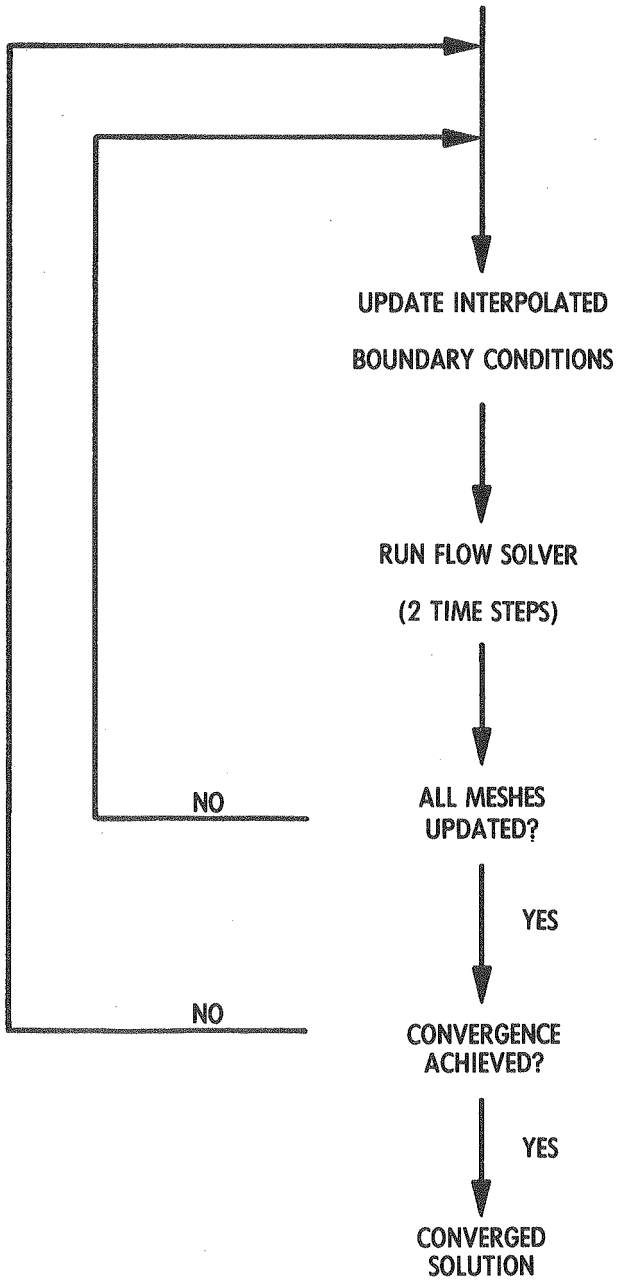


Fig. 16. XMER3D flowchart.

The wing surface is embedded within the fuselage mesh; hence, a hole surrounding the wing must be generated within the fuselage mesh. However, for most wing mesh topologies, the hole creation boundary within the wing mesh cannot be completely closed. As was indicated in Sec. 3.1, a hole creation boundary which is not completely closed may generate spurious hole points. It is therefore almost always necessary, when modeling wing/body junctures, to treat the hole generation process in a special manner.

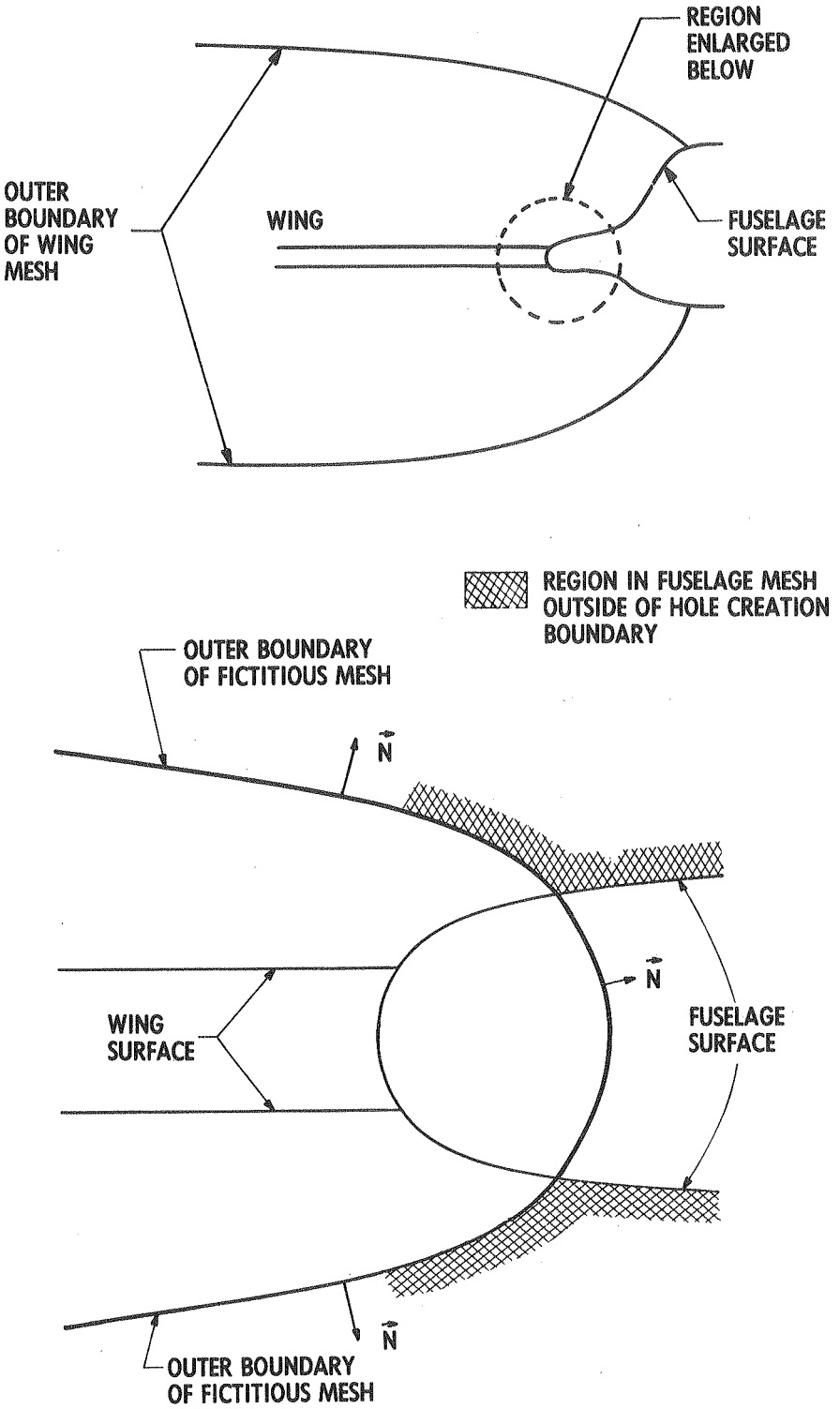


Fig. 17. Use of fictitious meshes at wing/body structure.

In Sec. 3.3.1, it was stated that a mesh which creates a hole in another mesh does not have to be the mesh which supplies interpolated flow-field information to the generated hole boundary points. It is therefore possible to generate "fictitious" meshes whose sole function is to create a satisfactory hole in another mesh, allowing a third mesh to supply interpolated flow-field information to the resultant hole boundary points. The use of a fictitious mesh to properly model a wing/body juncture is illustrated in Fig. 17. In the illustration, a boundary defined within the fictitious mesh completely encloses the wing, and therefore generates a hole in the body mesh for the wing; the wing mesh itself does not generate a hole at all.

Fictitious meshes have no other interaction with other meshes, i.e., no hole or outer boundary links to other meshes are defined. After processing by PEGSUS is completed, all fictitious meshes are discarded and therefore are not in any way processed by XMER3D.

5.2 MODELING THE INLET AND FUSELAGE. Wind tunnel test articles often include an active inlet which generally greatly affects the flow field in the vicinity of the fuselage. As a result, the computational methods employed for analysis must include the capability of modeling the inlet face and walls up to the engine face.

An example of an inlet/fuselage configuration is shown in Fig. 18. The configuration consists of two meshes. The first mesh models the fuselage; the second mesh models the inlet interior surfaces and a region ahead of the inlet. The inlet mesh passes through the inlet region of the fuselage mesh. The outer boundary of the inlet mesh forward of the inlet face receives flow field information interpolated from the appropriate mesh elements of the fuselage mesh. Solid wall boundary conditions are imposed on the interior walls of the inlet aft of the inlet face, and outflow conditions are imposed at the downstream boundary of the inlet mesh.

The surface of the fuselage mesh corresponding to the inlet face receives flow-field information interpolated from the interior of the inlet mesh. This is accomplished by defining the inlet face region as a boundary with a boundary link to the inlet mesh. The flexibility of PEGSUS allows any surface to be defined as an outer boundary. Therefore, the surface of the fuselage in the vicinity of the inlet/fuselage juncture is defined as an "outer" boundary with an outer boundary link to the inlet mesh. No holes are created in either mesh, since no solid surface from one mesh is embedded within the other mesh.

5.3 MODELING PYLONS AND STORES. An example of a typical fuselage, pylon, and store mesh configuration is depicted in Fig. 19. In this configuration, two stores are attached to the fuselage with pylons. The store and pylon solid surfaces are embedded within the fuselage mesh, and each pylon solid surface is embedded within its respective store mesh. Pylons are also attached to wing surfaces, but are not depicted in Fig. 19.

From a modeling standpoint, the relationship between a pylon and store (and between a pylon and fuselage or wing) is roughly equivalent to that between a wing and fuselage; in effect, a pylon is a wing of very small aspect ratio. As in the case of wing meshes, generating holes about pylon surfaces may be difficult unless fictitious meshes are employed. Typically, each pylon will have an associated fictitious mesh which generates a hole in the surrounding store and fuselage

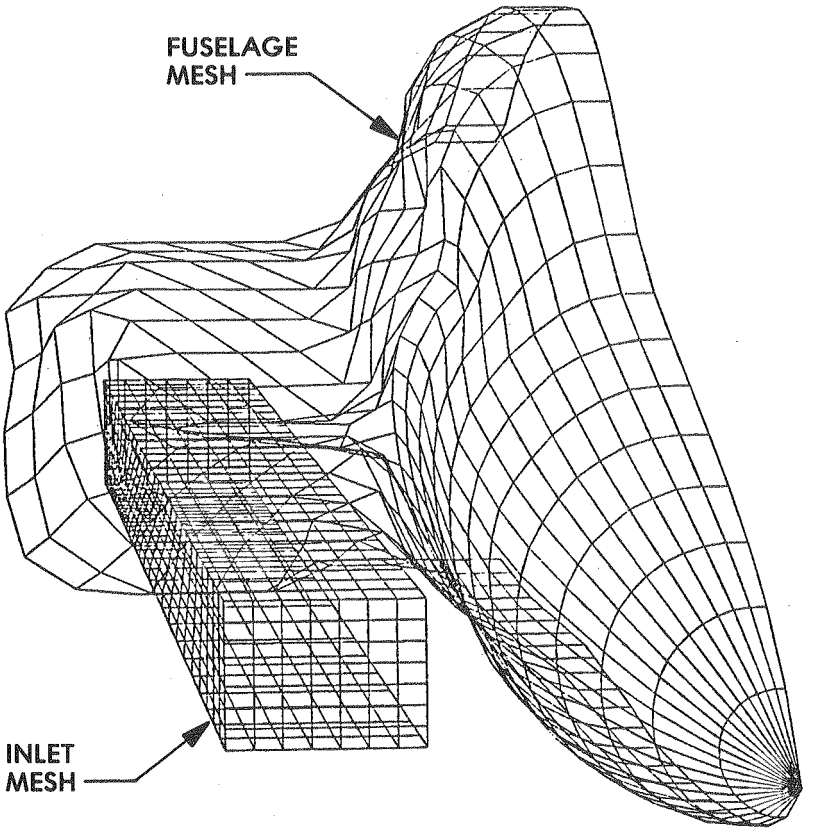


Fig. 18. Modeling of inlet/forebody configuration.

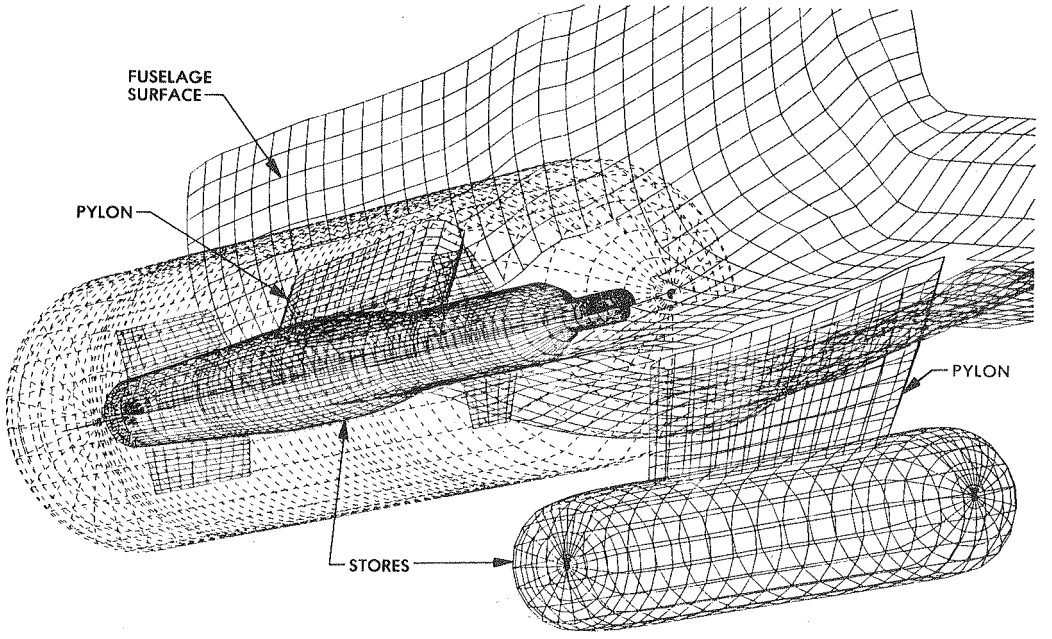
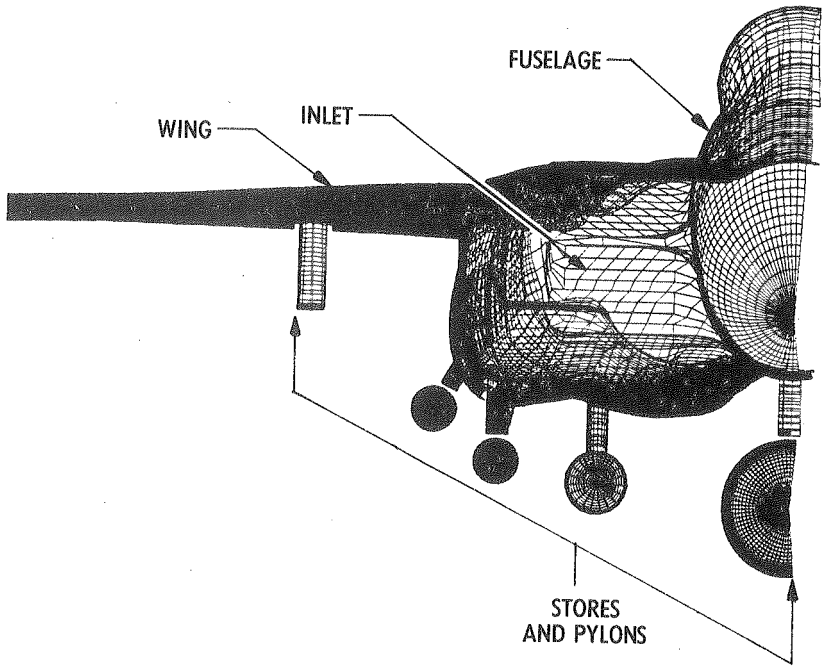


Fig. 19. Fuselage/pylon/store configuration.

(and/or wing) meshes. The resulting hole boundary points generated in the store and fuselage meshes can then be updated by elements within the pylon mesh. (The hole boundary points in the store mesh could also be updated by the fuselage mesh, and vice versa, if appropriate hole boundary links are defined).

Section 4.0 described how blanked mesh points are excluded from being updated by the XMER3D flow solver. Blanked points are usually either updated through interpolation, as in the case of hole boundary and outer boundary points, or are not updated at all, as in the case of hole points. However, blanked points may be updated explicitly through imposed boundary conditions. This allows XMER3D to accommodate arbitrary boundary conditions which are imposed on internal mesh points. In the case of store meshes, internal boundary conditions may be imposed to model thin fins. This is accomplished by designating the points corresponding to fin surface points as blanked points. Solid surface boundary conditions are then imposed on the blanked points. The flow solver will then automatically treat the blanked points as boundary points.

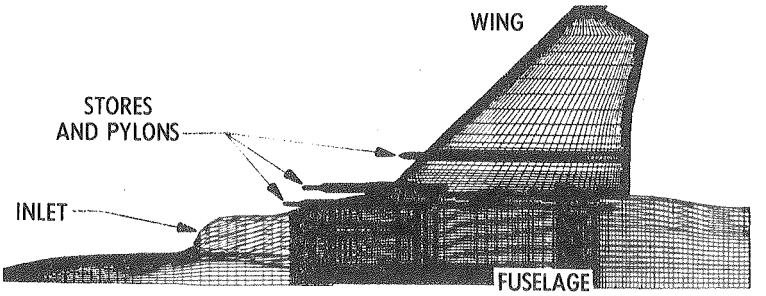
5.4 TOTAL CONFIGURATION. The solid surfaces of the most complex configuration modeled to date are depicted in Fig. 20. The configuration consists of a fuselage, wing, inlet, six pylons, and four stores. The complete configuration requires 50 meshes, of which twelve are fictitious. The composite mesh consists of a total of 1.33 million mesh points.



a. Front view

Fig. 20. Wing/body/store configuration.

5.5 SAMPLE CALCULATIONS. Euler equation solutions for the configuration of Fig. 20 are compared against experimental data in Fig. 21. Two plots of pressure coefficient are presented for a



b. Side view
Fig. 20. Concluded.

configuration angle-of-attack of 1.1 deg and free-stream Mach number of 0.98. Experimental data were obtained for longitudinal traverses beneath two model configurations. The leftmost plot depicts pressure coefficients for a configuration which included the fuselage, wing, inlet, and pylons, but did not include any stores. The right plot depicts pressure coefficients for the complete configuration depicted in Fig. 20.

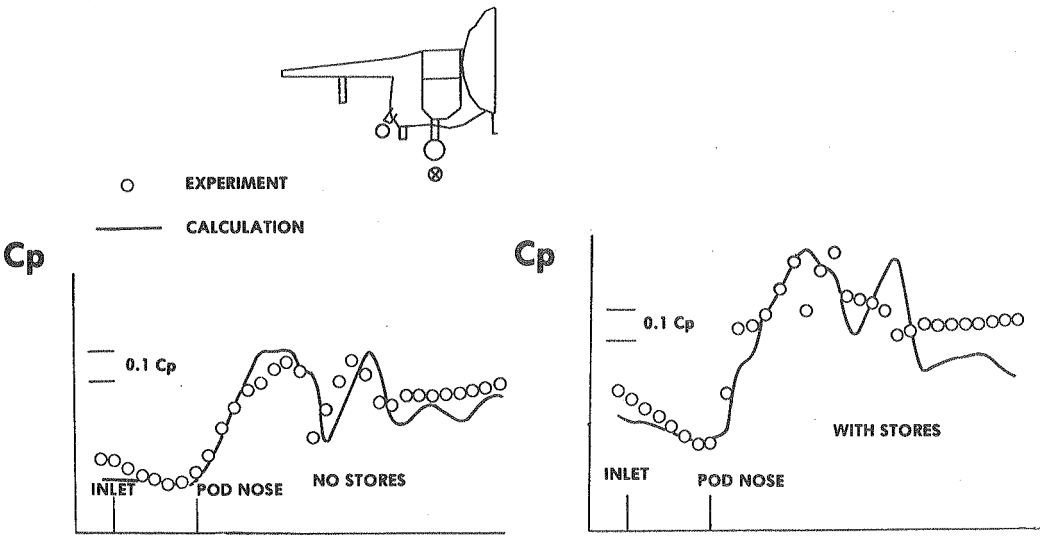


Fig. 21. Comparison of experimental and calculated pressure coefficients.

Comparisons between experimental and calculated pressure coefficients for the configuration which did not include stores were generally very good. The expansions and compressions within the flow field are shifted downstream somewhat, as is expected with inviscid solutions. The inviscid solutions also result in sharper peaks and valleys than are exhibited by the experimental data. The pressure comparisons for the configuration including stores are good, but are degraded somewhat in the wake region of the stores, as would be expected with the inviscid flow assumptions.

6.0 CONCLUSIONS. A computational scheme called chimera has been developed and implemented at the Arnold Engineering Development Center to provide the capability of performing computations on complex

aerodynamic configurations. The chimera scheme consists of two codes, PEGSUS and XMER3D. PEGSUS defines the communication among the meshes which comprise the aerodynamic configuration; XMER3D solves the Euler or Navier-Stokes equations for the multiple-mesh configuration. Chimera is presently being applied to aerodynamic configurations which include nearly complete aircraft definitions, including wings, inlets, pylons, and stores. The accuracy of the flow-field calculations is generally quite good, and validates the Euler solver used in conjunction with the domain decomposition approach as a predictor of flow fields about complex 3-D aerodynamic configurations. Future work includes the development of moving-mesh capabilities, which will allow the calculation of flow fields where solid bodies are moving relative to each other.

REFERENCES

1. J. A. BENEK, J. L. STEGER, and F. C. DOUGHERTY, "A Flexible Grid Embedding Technique with Application to the Euler Equations," AIAA 83-1944, July 1983.

2. J. A. BENEK, P. G. BUNING, and J. L. STEGER, "A 3-D Chimera Grid Embedding Technique," AIAA 85-1523, July 1985.

3. J. A. BENEK, F. C. DOUGHERTY, and P. G. BUNING, "Chimera: A Grid-Embedding Technique," AEDC-TR-85-64, December 1985.

4. J. A. BENEK, T. L. DONEGAN, and N. E. SUHS, "Extended Chimera Grid Embedding Scheme with Applications to Viscous Flows," AIAA 87-1126, June 1987.

5. F. C. DOUGHERTY, J. A. BENEK, and J. L. STEGER, "On Applications of Chimera Grid Scheme to Store Separation," NASA-TM-88193, October 1985.

6. N. E. SUHS, "Computations of Three-Dimensional Cavity Flow at Subsonic and Supersonic Mach Numbers," AIAA 87-1208, June 1987.

7. T. L. DONEGAN, J. A. BENEK, and J. C. ERICKSON, "Calculation of Transonic Wall Interference," AIAA 87-1432, June 1987.

8. T. H. PULLIAM, and J. L. STEGER, "On Implicit Finite Difference Simulations of Three-Dimensional Flows," AIAA 78-10, January 1978.

9. R. BEAM and R. F. WARMING, "An Implicit Factored Scheme for Compressible Navier-Stokes Equations," AIAA 77-645, June 1977.