

PEGASUS User's Guide

Version 5.1c
July, 2000

Norman E. Suhs and William E. Dietz

MICRO CRAFT

Stuart E. Rogers
NASA Ames Research Center



Steven M. Nash and Jeffrey T. Onufer
MCAT, Inc

PREFACE

Development of the automated grid overset approach implemented within the PEGASUS 5 code was supported by the High-Lift Subelement of the NASA Advanced Subsonic Technology / Integrated Wing Design Program. The technical point of contact was Dr. Karlin Roth at NASA Ames Research Center. Additional project support was provided by the Aeroscience and Flight Mechanics division at NASA Johnson Space Center.

TABLE OF CONTENTS

1.0 INTRODUCTION

2.0 GENERAL APPROACH

2.1 Automatic Minimum Hole Cutting

2.2 Automation of Outer Boundary Specifications

2.3 Interpolation

2.4 Level 2 Interpolation

2.5 Orphan Point Fixing

2.6 Projection

2.7 Unblanking

2.8 Restarting

3.0 PROGRAM IMPLEMENTATION

3.1 Compilation

3.2 User Inputs

3.3 Utility Codes

3.4 Output File Formats

3.5 Directory Structure

4.0 USING PEGASUS

4.1 Setting Up the Input

4.2 Running PEGASUS

4.3 Using Externally Generated Hole Definitions

4.4 Operational Hints

4.4.1 Gaps in Surfaces

4.4.2 Thin Trailing Edges

4.4.3 Projection

5.0 REFERENCES

6.0 APPENDICES

A Glossary

B Example of PEGASUS Input File

C Obsolete Inputs

PEGASUS 5 User's Guide		
Previous Section	Return to Table of Contents	Next Section

Section 1: INTRODUCTION

PEGASUS 5.1 is the latest version of the PEGASUS series of mesh interpolation codes. It is a fully three-dimensional code. The main purpose for the development of this latest version was to significantly decrease the number of user inputs required and to allow for easier operation of the code. This guide is to be used with the user's manual for version 4 of PEGASUS (Ref. 1). A basic description of methods used in both versions is described in the Version 4 manual. A complete list of all user inputs used in version 5.1 is given in this guide.

The major new features are as follows:

- The minimum holes can be automatically cut if the boundary conditions for each mesh are supplied.
- Growth of the minimum holes is achieved through "level 2" interpolation. In level 2 interpolation, interpolated boundary points receive information from the finest mesh available in overlapping regions. As a result, link lists (required in Version 4) are no longer used.
- Outer boundaries are automatically specified if the boundary conditions for each mesh are supplied.
- Memory is dynamically allocated at run time. The code is written almost entirely in Fortran 90, and memory allocation is achieved through Fortran 90 constructs. Some subroutines are written in Fortran 77, which has been designated as a standard subset of Fortran 90. There is no C code used in PEGASUS 5.1.
- Restarting is similar to the UNIX "make" facility. As new meshes or inputs are added, PEGASUS will recognize these new changes and performs only the required operations.
- Existing hole definitions contained in IBLANK records of PLOT3D grid files can be used to define the holes in PEGASUS 5.1. This allows stand-alone hole-cutting methods to be employed if desired. In addition, legacy grid systems can be used with existing hole definitions.

Complete automation the overset mesh interpolation process of an arbitrary set of meshes is a very difficult task. If automation is not achieved with the methods supplied in PEGASUS 5.1, the standard "manual" hole cutting and outer boundary specification inputs from PEGSUS 4 are still available.

PEGASUS 5 User's Guide

[Previous Section](#)

[Return to Table of Contents](#)

[Next Section](#)

PEGASUS 5 User's Guide		
Previous Section	Return to Table of Contents	Next Section

Section 2: GENERAL APPROACH

The new features that have been added to PEGASUS 5 are discussed in this section.

2.1 Automatic Minimum Hole Cutting

An automatic hole-cutting capability is provided with PEGASUS 5.1 In many cases, a good hole structure can be developed with no intervention by the user. However, the hole generation process can be manipulated through user inputs if necessary.

The hole generation process requires a definition of the solid boundaries of the configuration. This information is supplied in the standard input file. See the \$BCINP namelist for a description of this input and Appendix B for an example. This file can either be created manually using an editor, or can be generated automatically (with possibly some modifications necessary) if OVERFLOW input is available.

The automatic hole generation process is illustrated using a two-dimensional example (i.e., a 3-element airfoil, used throughout this manual to illustrate aspects of PEGASUS operation). Figure 1 illustrates the solid boundaries of the airfoil, as defined in the boundary condition file.

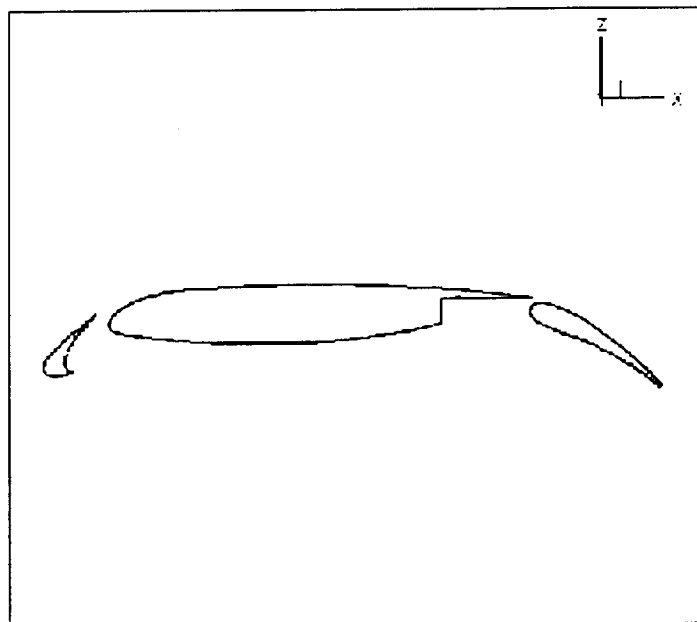


Figure 1. Three-Element Airfoil Solid Boundaries

The entire point of the hole-generation process is to partition the region surrounding the airfoil into "inside" and "outside" regions. In PEGASUS 5.1, this is accomplished using Cartesian meshes, where it is desired to mark each Cartesian element as an "inside", "outside", or "fringe" element. In the simplest case, a single Cartesian mesh is generated which contains the solid boundaries of the configuration. The elements of the Cartesian mesh which intercept the solid surface elements of the airfoil are found: these are designated as fringe elements. Some of the fringe elements in the slat-wing region of the 3-element airfoil are depicted in Figure 2.

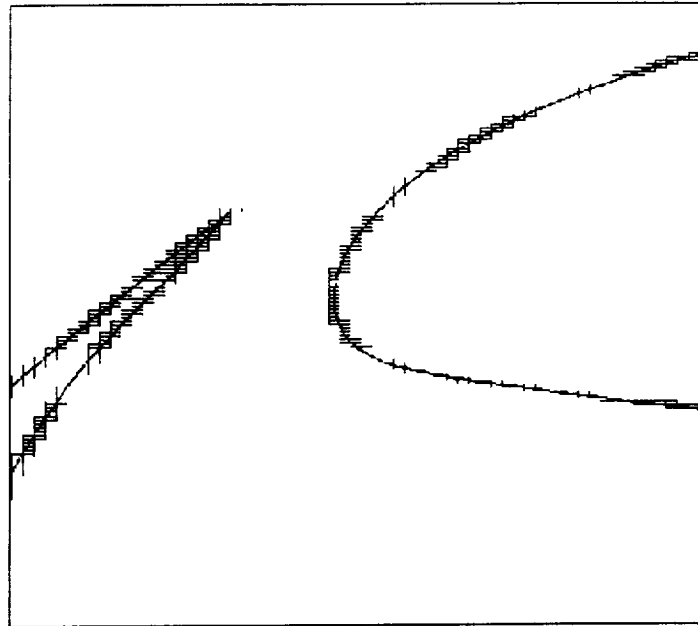


Figure 2. Fringe Elements

It is assumed that the corner elements of the Cartesian mesh are "outside" elements. Any unidentified (i.e., non-fringe) element that is adjacent to an outside element must itself be an outside element. The outside region is then identified by marching 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 Figure 3. Finally, any element remaining that is not either an outside or fringe element must be an "inside" element. Inside elements are depicted in Figure 4.

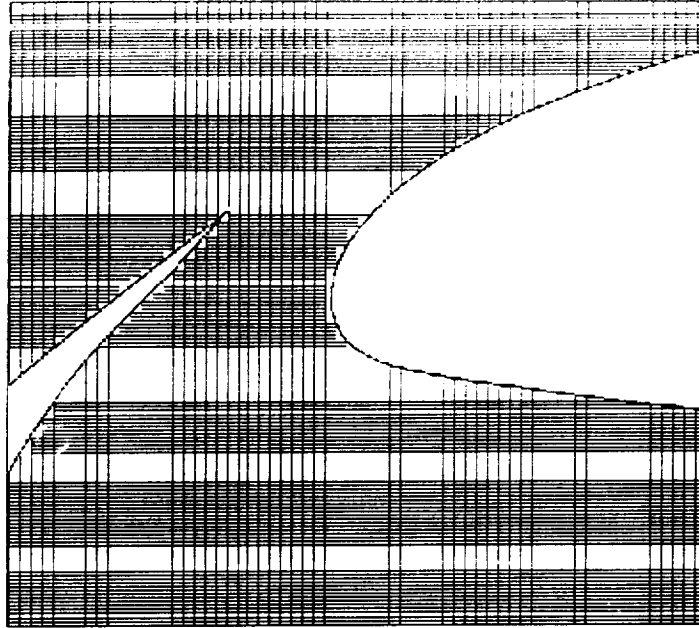


Figure 3. Outside Elements

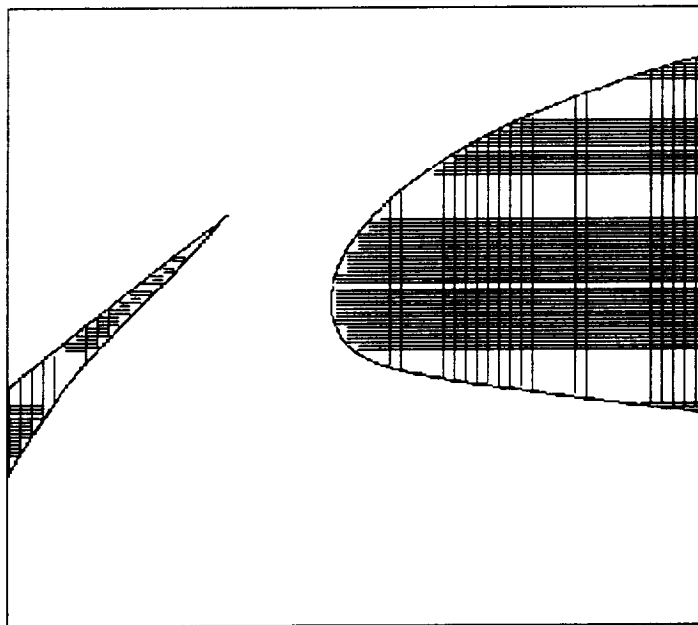


Figure 4. Inside Elements

The Cartesian mesh is now a completed "hole map". Given an arbitrary grid point, the Cartesian element within the hole map in which the point resides can very quickly be identified. Points that are found to be encompassed by "outside" or "inside" elements are marked as field points or hole points, respectively. Points that fall within fringe elements must undergo further processing. PEGASUS 5.1 uses a "line-of-sight" algorithm to determine the status of an arbitrary grid point; this algorithm simply states

If a clear line-of-sight exists between the point and an outside or inside (i.e., non-fringe) element, the point will assume the identity of that element. A clear line-of-sight means that a vector from the point to a neighboring non-fringe element does not intersect the solid surface contained in the fringe element. The algorithm is illustrated in Figure 5. Points that can "see" an outside element are field points; points that can "see" an inside element are hole points.

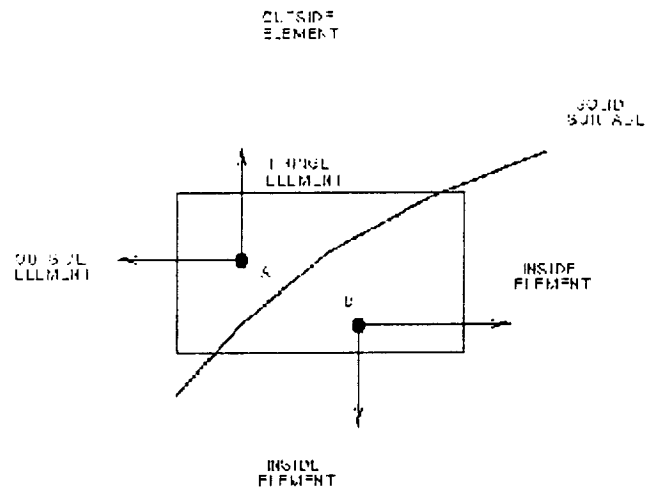


Figure 5. Line-of-Sight Algorithm

In the example of Figure 5, Point A is outside. Point B is inside, and will be marked as a hole point. Points that cannot be identified are considered outside points.

2.2 Automation of Outer Boundary Specifications

The automation of the specification of outer boundaries is relatively straightforward. If all boundary conditions have been specified for a mesh, the minimum and maximum index surfaces that have not been specified as boundary conditions for the flow solver are designated as the outer boundaries. Outer boundaries are assumed to receive boundary information via interpolation from other meshes.

To automate the outer boundary specifications within PEGASUS, the boundary conditions for each grid must be available as input to PEGASUS. The boundary conditions are entered into PEGASUS using the \$BCINP namelist in the standard input file. By default, PEGASUS will determine outer boundaries for all meshes.

The manual approaches for specifying both outer boundaries and interior blanked regions are still acceptable inputs to PEGASUS. These inputs can still be used separately or in combination with the new method.

In PEGASUS version 4, the user specifies whether a hole boundary is to have a single or double fringe

of by specifying points with a single input. The fringe level for outer boundaries, on the other hand, had to be explicitly specified. Now, with outer boundaries automatically specified, the fringe level for outer boundaries can be set with a single input. The user can set the fringe level for holes and outer boundaries with a single input, or set the fringe level for holes and outer boundaries separately. It should be noted that not all flow solvers can handle a mixed single/double fringe level of interpolated boundary points.

2.3 Interpolation

In version 4, the LINK list input is a priority list containing the names of meshes that are to be searched for interpolation stencils that can provide information for interpolated boundary points. The meshes in the priority list are searched in the order in which they appear in the input. If a legal interpolation stencil is found, then the processing for that interpolated boundary point is discontinued. If no legal interpolation stencil can be found in the first mesh in the priority list, then the second mesh is searched for an interpolation stencil. The process continues until either (1) interpolation stencils have been found for all interpolated boundary points, or (2) the mesh priority list is exhausted. Any interpolated boundary points that have no legal interpolation stencil in any of the meshes in the priority list are termed "orphans."

The user typically places the finer meshes at the top of the priority list and coarser meshes at the bottom of the list. This method usually gives adequate results but still leaves the user with only visualization techniques to evaluate the size difference between the interpolated boundary point cell and the interpolation stencil. (An interpolated boundary point cell is the cell adjacent to the interpolated boundary point in the positive J, K, and L directions.) It is assumed here that to obtain the "best" interpolation, the interpolated boundary point cell and interpolation stencil cell must have similar sizes. Use of the link-list approach, although simple to implement and computationally efficient, may not find the "best" pairing of an interpolated boundary point cell and an interpolation stencil. For example, if a user specifies a coarse mesh at or near the top of the link list, a significant difference in size may be found between the interpolated boundary point cell and the interpolation stencil.

Automation of this process must therefore take into account the sizes of the interpolated boundary point cell and the interpolation stencil when determining the "best" pairing. Additionally, the user needs to receive diagnostic information to determine the worth of the interpolated boundary points-interpolation stencil pairs.

The interpolation stencil that is determined to be the "best" for an interpolation boundary point is based on quality and the cell difference parameter. Quality is defined as a measure of the information that is being interpolated from non-hole points. As in previous versions of PEGASUS, if one of the stencil points is a hole point the interpolation stencil is not accepted. Quality is computed by first creating a temporary cell with the values at each of the eight points set equal to 1.0 (the point is a field point) or 0.0 (the point is a boundary point). Quality is then determined by performing a tri-linear interpolation using the computed interpolation coefficients. If all of the stencil points are field points, quality will have a value of 1.0. If for example, one of the stencil points is an interpolated boundary point and the interpolation coefficients are 0.5, 0.5, and 0.5, and, quality would be equal to 0.875.

The cell difference parameter (CDP) is defined as follows:

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

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

V_B = the volume of the boundary cell

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

V_I = the volume of the interpolated cell

Values for the cell difference parameter will vary from 0 (the best) to "big." It is yet to be determined what is "too big." Examples of the cell difference parameter are shown in Figure 6 for 2-dimensional cells.

















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

Figure 6. Examples of Cell-Difference Parameter

Quality and the cell difference parameter are used to determine from which interpolation stencil, and therefore, from which mesh, an interpolated boundary point receives its data. Figure 7 depicts this logic. The quality value is checked first. If the quality is less than a user specified cutoff level (QCUTOFF), the interpolation stencil is no longer considered.

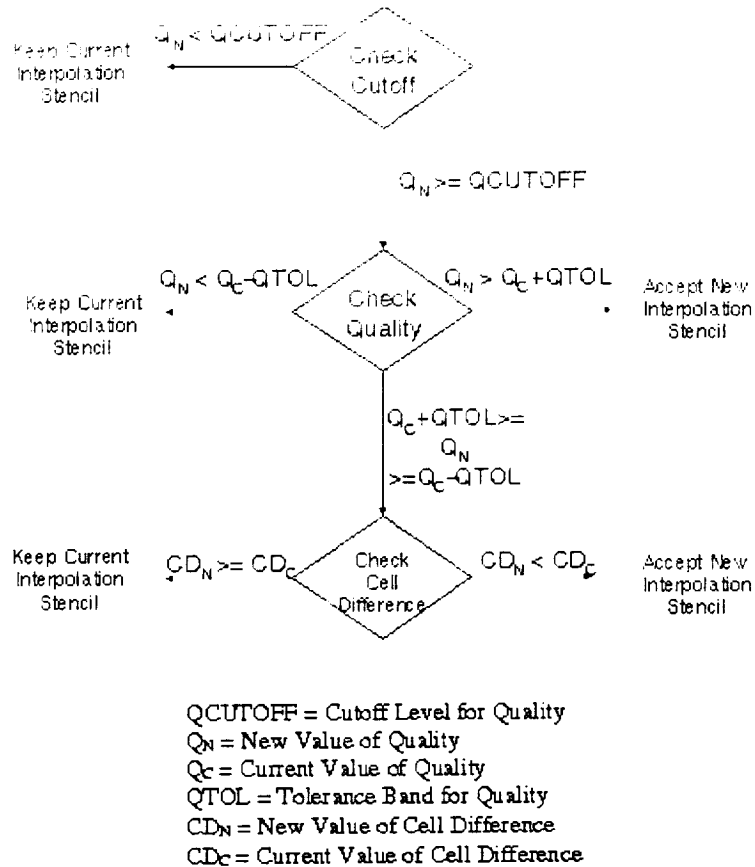


Figure 7. Logic for Interpolation Stencil Choice

If the quality of the interpolation is better than a previously selected interpolation stencil by some tolerance (QTOL), the newer interpolation stencil is selected. If the value of quality is within the tolerance value, the interpolation stencil with the lesser value for cell difference is selected. If the computed quality is less than the current quality minus the tolerance, the previously accepted interpolation stencil is kept.

2.4 Level 2 Interpolation

Once the minimum hole has been established, the hole typically must be enlarged to improve the communication among the meshes. When only a minimum hole is cut, the fringe about the hole often causes very coarse mesh points to interpolate from very fine mesh elements. This disparity in mesh resolutions can severely impact accuracy, especially in viscous solutions, where a coarse mesh might be attempting to interpolate flow field information from within a boundary layer. In addition to being able to establish blanked regions in meshes with solid boundaries, it is also a requirement to have the capability to create holes with refinement meshes that are added to the domain. This cannot be accomplished through the usual hole-cutting methods, since refinement meshes typically do not have solid walls that would be used for automatic hole cutting.

In order to handle these requirements a new approach for creating the interpolation boundary was developed. This approach, termed "level 2" interpolation, has the basic philosophy that coarser mesh points are interpolated from finer mesh points wherever possible. If more than two meshes are present, the coarser mesh will interpolate from the medium mesh, which will interpolate from a finer mesh. For all interpolations of this type quality is kept equal to 1.0. This approach results in holes that are enlarged only if better communications are obtained by doing so. Therefore, the flow field will be computed for a given region only by the mesh with the best resolution. In addition, proper communications is established by this approach when refinement meshes are added. Figure 8 depicts the hole structure in the outer mesh in the 3-element airfoil case. Note that the outer boundary of the wing mesh has also been moved in so that the outer boundary of the wing mesh obtains information from mesh elements of similar refinement.

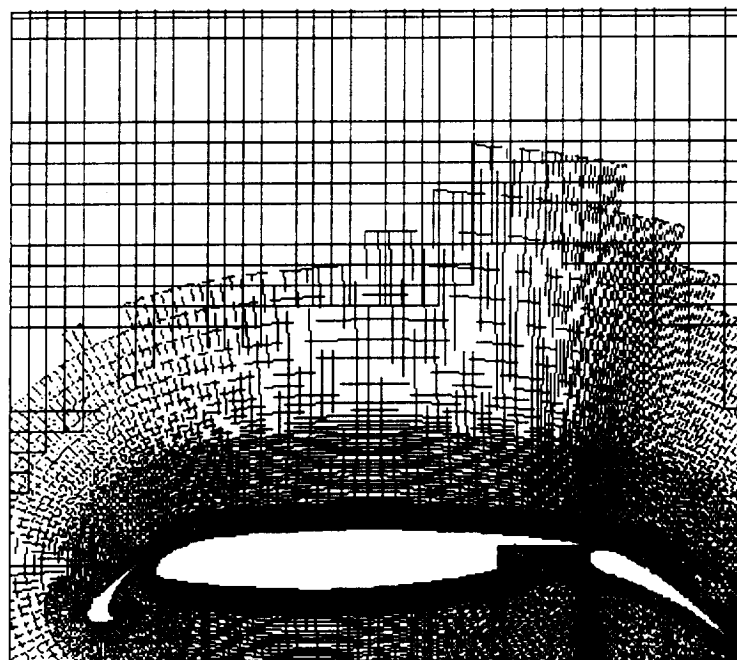


Figure 8. Hole Structure Due to Level 2 Interpolation

Level 2 interpolation is not initiated within PEGASUS until all boundary creation (hole cutting and outer boundary definitions) is completed.

To implement level 2 interpolation, PEGASUS first attempts to interpolate every point in each mesh from all other meshes. As a result, PEGASUS performs far more point interpolations than did earlier versions of PEGASUS. Therefore, it is critical that the interpolation process in PEGASUS be as efficient and robust as possible. As in earlier versions of PEGASUS, Version 5.1 uses a Newton's iteration

method of determining interpolation coefficients for each point. Newton's methods are very efficient if a good initial guess is available. Earlier versions of PEGASUS used an exhaustive search on a sparse representation of the donor grid to provide initial guesses to the Newton's method. It was determined that in some cases PEGASUS was spending large amounts of computational time determining these initial guesses, particularly since even a sparse representation of a large grid can still represent a significant effort when using an exhaustive search.

PEGASUS 5.1 uses data structures (Alternating Digital Trees, or ADTs, Ref. 2) for each mesh to determine an approximate closest point in the donor mesh to the target boundary point in the recipient mesh. Use of the ADT is extremely fast; being a tree structure, the ADT approach can handle very large meshes without a significant increase in the amount of time required to find the approximate closest point. Once the approximate closest point is found, the Newton's method is used to iterate to the final stencil reference point and compute the associated interpolation coefficients. PEGASUS also uses the ADTs as a means to cross boundaries (e.g., cut planes) during the Newton's method iteration without requiring knowledge of the mesh topology.

2.5 Orphan Point Fixing

An additional option is offered to the user that provides the ability to change second fringe "orphan" points back to field points. This option should only be used with flow solvers that can handle a mixture of single and double fringes along any interpolated boundary (e.g., OVERFLOW (Refs. 3-5) or INS3D (Ref. 6)). It should be noted that fixing a second fringe orphan only changes that point to a field point and does not affect the quality of any interpolation point that uses this second fringe orphan point as an interpolation stencil. Also, even if a second fringe orphan point is fixed, it is still retained as an orphan point within PEGASUS and efforts will be made in subsequent restarts to find a legal interpolation stencil for this point.

2.6 Projection

Problems have often arisen when two or more grids overlap at a region near a solid wall, particularly with the small spacing found in meshes generated to resolve boundary layers. This difficulty is due to the discretization of curved surfaces. Points that are on the actual surface can be found to be inside the discretized solid boundary of another surface. To correct this problem, mesh projection methods have been used successfully. Programs like PROGRD (Ref. 7) move the mesh points in one mesh relative to another mesh. Once the grids have been projected and stored, interpolation coefficients may be found by PEGASUS in the normal manner.

PEGASUS 5.1 has a built-in and highly automated mesh projection capability, which is based on the PROGRD algorithm. For a given mesh, all other overlapping meshes are projected onto its solid surfaces. The interpolation process then uses these projected mesh definitions to determine interpolation coefficients. PEGASUS does not modify the grids that will be used by the flow solver. However, each

Some large point will receive information from interpolation elements from the temporary projected mesh definitions. Mesh projection can be controlled through the user inputs **DISTANCE**, **ANGLE**, **SHIFT**, **PINCLUDE**, **PEXCLUDE** in the \$GLOBAL and \$MESH namelists.

2.7 Unblanking

A feature that has been added to aid in cleaning up hole cutting is the unblanking feature. This feature is a manual operation that uses the \$REGION/\$VOLUME user inputs. Points of a specified mesh can be modified so as not to be hole points. This procedure is performed immediately after hole cutting. Points that are to be unblanked will become either a field or interpolated boundary point.

2.8 Restarting

In the previous version of PEGASUS, restarting was handled in a very labor-intensive manner. Each new restart required a new user input file that specified the changes, as well as the relationship, of the new added meshes to the previously added meshes. This operation was prone to errors and required a new approach, since the number of meshes and connections would increase prohibitively. The new approach to restart requires no special knowledge of what has occurred in previous executions. The user is required only to modify the standard input file or the meshes, as needed. PEGASUS automatically determines what work needs to be performed to complete the PEGASUS execution. In this manner, restarting in PEGASUS is very similar to using the UNIX "make" utility.

The procedures in PEGASUS version 5.1 have been written so create the communication data in a path independent manner. Path independence was not the case with restarting for PEGASUS 4.

In order to restart a PEGASUS job from the beginning, simply remove or rename the **WORK** directory, for example:

```
/bin/rm -rf WORK
```

PEGASUS 5 User's Guide		
Previous Section	Return to Table of Contents	Next Section

PEGASUS 5 User's Guide		
Previous Section	Return to Table of Contents	Next Section

Section 3: PROGRAM IMPLEMENTATION

3.1 Compilation

PEGASUS will usually be obtained in a "gzipped" tar file. When the file is unzipped and extracted, the top directory will contain a configure script and several subdirectories. The **/peg** subdirectory contains the source code, and the **/tools** subdirectory contains several PEGASUS utilities that aid in setting up the input, plotting, file conversion, and diagnostics.

PEGASUS has been developed and tested on SGI and Cray platforms. Every attempt has been made to conform to standard Fortran 90 conventions; therefore, compilation on other platforms with compliant Fortran 90 compilers should be straightforward.

To compile and install PEGASUS and all of the utility programs and scripts, follow these steps:

1. Run the configure script. Type "configure -help" to see a list of options. Examples include:

configure --enable-dp	To enable double-precision
configure --enable-mips2	SGI -mips2 compiler option
configure --enable-mips3	SGI -mips3 compiler option
configure --enable-mips4	SGI -mips4 compiler option
configure --enable-debug	To enable debug compiler flags and output
configure --installdir=DIR	Executables will be installed in directory DIR
2. Compile: type "make"
3. Install: type "make CMD=install"

The executable will be named **pegasus5**.

It is recommended that PEGASUS be compiled and run in double precision when viscous grids are used. Running viscous cases in single precision has been seen to sometimes adversely affect the hole-generation process, and in some cases can affect interpolation as well.

3.2 User Inputs

Two types of input must be supplied: the volume grids, and the standard input file. There are some utility codes, which are described in Section 5.3, which help in the preparation of this input. The details of these inputs are described in this section.

The individual volume grids are stored in a sub-directory within the project directory called **X_DIR**. See Section 3.5 for a description of the directory structure used by **PEGASUS5** code. The grids are to be stored as unformatted, single-zone, PLOT3D grid files. These grid files are named "mesh_nameA.x", "mesh_nameB.x", etc, where mesh_nameA, mesh_nameB, ... correspond to the NAME variables in the \$MESH namelists in the standard input file.

All of the user inputs to **PEGASUS** are read from the standard input. All inputs are in a form similar to FORTRAN NAMELISTS. The following is a list of all of the NAMELISTS defined in **PEGASUS**. Not all of these NAMELISTS have to be included in the input, and the order in which the NAMELISTS appear in the input does not matter. Most problems will only require input to be specified in the first three of these NAMELISTS:

- \$GLOBAL
- \$MESH
- \$BCINP
- \$HCUT
- \$OUTER
- \$BOUNDARY
- \$SURFACE
- \$BOX
- \$REGION
- \$VOLUME
- \$LEVEL2

Note that all user input record names are capitalized and all alpha-numeric names (i.e., NAME, ISPARTOF) are case sensitive. Examples of some these inputs is given in Appendix B. Obsolete user inputs (those used in version 4.x but not in version 5.1) are listed in Appendix C.

In the namelists which require grid indices, negative numbers can be used to specify indices relative to the last index. For example, an index of $j=-1$ is equivalent to $j=j_{max}$, $j=-2$ is equivalent to $j=j_{max}-1$, etc.

\$GLOBAL

This input sets global values for all meshes. Each record can be overridden for any individual mesh by using the \$MESH input. However, the mesh override is limited to that individual mesh. For subsequent meshes, the global values are used unless specifically overridden.

Variable	Description	Default

ANGLE	Maximum angle, in degrees, between the surface normal of the projected point and the reference surface of projection. If greater than this angle, the projection is not allowed.	30.0
CARTX CARTY CARTZ	<p>Each of these inputs is a pair of numbers to set the min and max boundaries in the x, y, or z directions of the Cartesian boxes used in the automatic hole cutting. If they are left at their default values, the hole-cutting cartesian boundaries are set to extend just beyond the surfaces making up the hole cutter.</p> <p>This input can be used to limit the extent of the of the volume in which holes can be blanked out by the hole-cutters, and in some instances, can be used to help close off leaks in a hole-cutting surface.</p>	-1.e30, 1.e30
CENTER	Set equal to .TRUE. to create an internal cell-centered grid.	.FALSE.
CNX, CNY, CNZ	Integer dimensions of Cartesian grids (hole maps) used for hole cutting.	512
DISTANCE	A positive integer value that controls the maximum distance that a surface point can be projected. The input DISTANCE is the number of grid cells in the solid-wall normal direction used to compute the actual distance limit.	20
EPS	Tolerance of interpolation coefficient. Value of interpolation coefficients less than 0 and greater than 1.0 that is allowed for a converged stencil.	0.001
EXCLBC	Set equal to .TRUE. if the points specified as boundary conditions are to be excluded as interpolated boundary points.	.TRUE.

FRINGE	Set equal to 1 for single fringe or equal to 2 for double fringe.	1
HBFRNG	Hole boundary fringe. Set equal to 1 for single fringe or equal to 2 for double fringe.	Value of FRINGE
HOLE1ST	If set equal to .TRUE., hole cutting, both manual and automatic, are performed prior to interpolation.	.FALSE.
HCUT	Set equal to .TRUE. if automatic hole cutting is to be performed.	.TRUE.
INCORE	Specifies if main data (e.g., meshes, interpolation values, etc.) are kept incore (=TRUE.) or written and read from file (=FALSE.)	.FALSE.
LEVEL2	Set equal to .TRUE. if "level 2" interpolation is to be performed.	.TRUE.
OBFRNG	Outer boundary fringe. Set equal to 1 for single fringe or equal to 2 for double fringe. If this value is not specified it takes on the value of FRINGE.	Value of FRINGE
OCTLEVEL	Integer level of octree refinements used to define automatic hole cutting boundaries.	9
OFFSET	Set to a positive integer value, zero or greater. This value specifies the increase of the minimum hole that was created by an automatic hole cutter. For example, if OFFSET=2, all field points within 2 indices of the hole are changed to hole points.	0

ORPHFIX	Set equal to .TRUE., second fringe orphan points will be fixed, i.e., returned to field values. Certain flow solvers (e.g., OVERFLOW and INS3D) allow for mixed fringed conditions (single and double) and can use this option.	.TRUE.
OUTER	Set equal to .TRUE. if automatic outer boundary specifications are to be performed.	.TRUE.
THRU	<p>Perform processing up to and including a particular operation. The following are the operations in the order that they are executed:</p> <p>projection - project meshes with solid boundary conditions</p> <p>octree - creates a search tree for each mesh</p> <p>interpolate - interpolates each mesh-to-mesh combination</p> <p>auto_hbound - creates automatic Cartesian hole cutters</p> <p>man_hboun - creates surface cutter based on user inputs</p> <p>auto_cut - cuts holes using automatic Cartesian hole cutter</p> <p>man_cut - cuts holes using manual hole cutting</p> <p>comp_hole - assembles composite holes</p> <p>spec_int1 - specify points for "level 1" interpolation</p> <p>spec_level1 - pick "best" points for "level 1" interpolation</p>	xintout

	<p>spec_int2 - specifies points for "level 2" interpolation</p> <p>spec_level2 - pick "best" points for "level 2" interpolation</p> <p>xintout - produces an XINTOUT file for current PEGASUS execution (perform all operations)</p>	
PROJECT	Set equal to .TRUE. if projection of solid wall boundary surfaces is to be preformed.	.TRUE.
SHIFT	This input is the same as the shift parameter in PROGRD and is used only if PROJECT=.TRUE.. If equal to 1, move all points in J/K/L direction by same increment as surface point. If equal to 2, move nearest half of point in J/K/L direction by same increment as surface point. If equal to 3, hold opposite boundary fixed in J/K/L direction and redistribute interior points proportional to arclength distance from wall.	3
QCUTOFF	The level of quality that is accepted for a mesh. Interpolated boundary points with a value of quality below this level will be considered an "orphan" point and no interpolation information will be supplied.	0.0
QTOL	Tolerance band for quality. This value is used when selecting the "best" interpolated boundary point cell-interpolation stencil pair (see Section 2.3 and Figure 7).	0.01
XINCLUDE, YINCLUDE, ZINCLUDE	Range of X, Y, and Z values, respectively, to be included as boundary points. Composed of two values (a minimum and a maximum value) in each of the X, Y, and Z directions, respectively.	none

\$MESH

Variable	Description	Default
QTOL	Tolerance band for quality for a. This value is used when selecting the "best" interpolated boundary point cell-interpolation stencil pair (see Section 2.3 and Figure 7).	0.01 or QTOL specified in \$GLOBAL
QCUTOFF	The level of quality that is accepted for a. Interpolated boundary points with a value of quality below this level will be considered an "orphan" point and no interpolation information will be supplied.	0.0 or the value of QCUTOFF in \$GLOBAL
HBFRNG	Hole boundary fringe. Set equal to 1 for single fringe or equal to 2 for double fringe.	The value of the next fringe value specified in the ascending order: FRINGE for this mesh, HBFRNG for \$GLOBAL, FRINGE for \$GLOBAL
OBFRNG	Outer boundary fringe. Set equal to 1 for single fringe or equal to 2 for double fringe.	The value of the next fringe value specified in the ascending order: FRINGE for this mesh, OBFRNG for \$GLOBAL, FRINGE for \$GLOBAL

ORPHFIX	Set equal to .TRUE., second fringe orphan points will be fixed, i.e., returned to field values. Certain flow solvers (e.g., OVERFLOW and INS3D) that allow for mixed fringed conditions (single and double) can use this option.	.FALSE.
PHANTOM	Setting PHANTOM=.TRUE. denotes this mesh is a "phantom" mesh, to be used only for manual hole cutting, and is not part of the final grid system.	.FALSE.
ALFA, BETA, GAMA	Euler angles (degrees) determining rotation of input meshes. ALFA, BETA, and GAMA are rotations about the Z, Y, and X coordinates, respectively. The coordinates of a mesh are transformed in the following sequence of operations: (1) scaling, (2) translation, and (3) rotation.	0.0
EPS	Tolerance of interpolation coefficient. Value of interpolation coefficients less than 0 and greater than 1.0 that is allowed for a converged stencil.	Value of EPS in \$GLOBAL
FRINGE	Set equal to 1 for single fringe or equal to 2 for double fringe.	Value of FRINGE in \$GLOBAL
JINCLUDE, KINCLUDE, LINCLUDE	Range of J, K, and L indices, respectively, to be included for boundary points. Composed of two values (a starting and an ending value) in each of the J, K, and L directions, respectively.	none

EXCLBC	Set equal to .TRUE. if the points specified as boundary conditions are to be excluded as boundary points.	.TRUE.
NAME	Name of mesh to which NAMELIST record refers (must be any alpha-numeric characters with no special symbols). A NAME is a string not exceeding ICHAR characters.	none
SCALE	Mesh scaling factor. The original mesh coordinates are multiplied by the scaling factor. The coordinates of a mesh are transformed in the following sequence of operations: (1) scaling, (2) translation, and (3) rotation.	1.0
XINCLUDE, YINCLUDE, ZINCLUDE	Range of X, Y, and Z values, respectively, to be included for boundary points. Composed of two values (a minimum and a maximum value) in each of the X, Y, and Z directions, respectively.	none
XR,YR,ZR	Point (in translated and scaled coordinates) about which input mesh is to be rotated. The coordinates of a mesh are transformed in the following sequence of operations: (1) scaling, (2) translation, and (3) rotation.	0.0

X0.Y0.Z0	Mesh translation factors. These factors are added to the original mesh coordinates. The coordinates of a mesh are transformed in the following sequence of operations: (1) scaling, (2) translation, and (3) rotation.	0.0
OFFSET	Set to a positive integer value, zero or greater. This value specifies the increase of the minimum hole that was created by an automatic hole cutter. For example, if OFFSET=2, all field points within 2 indices of the hole are changed to hole points.	0
PROJECT	Logical variable which controls whether or not solid-wall points in this mesh will actively project themselves onto other solid-wall surfaces in other meshes. Even if this is set to .FALSE., this meshes solid walls may be used as a reference surfaces for other meshes to project onto, unless this mesh is listed in the PEXCLUDE list for all other meshes.	Value of \$GLOBAL PROJECT.

SHIFT	This input is the same as the shift parameter in PROGRD and is used only if PROJECT=.TRUE.. If equal to 1, move all points in J/K/L direction by same increment as surface point. If equal to 2, move nearest half of point in J/K/L direction by same increment as surface point. If equal to 3, hold opposite boundary fixed in J/K/L direction and redistribute interior points proportional to arclength distance from wall.	3
ANGLE	Maximum angle between the surface normal of the projected point and the reference surface of projection. If greater than this angle, the projection is not allowed.	30.0
DISTANCE	A positive integer value that controls the maximum distance that a surface point can be projected. The input DISTANCE is the number of grid cells in the solid-wall normal direction used to compute the actual distance limit.	20
PINCLUDE	Names of meshes that are to be included for projection with this mesh.	All meshes.
PEXCLUDE	Names of meshes that are to be excluded from projection with this mesh.	No meshes.

\$BCINP

This input is like the BCINP namelist in the **OVERFLOW** flow solver. This namelist specifies all of the

boundary conditions for each mesh. PEGASUS uses this information to automatically calculate (based on the specification of the solid walls), and to automatically determine the outer boundary points which will require interpolation. It also uses these boundary conditions to determine certain mesh topologies, such as o-grid periodic meshes, 2D meshes, and symmetry planes.

Variable	Description	Default
ISPARTOF	Name of MESH for which these boundary conditions apply.	None
IBTYP	<p>List of boundary condition types</p> <p>1 - 8 = Solid wall</p> <p>10 = O-grid periodicity (apply to first or last plane)</p> <p>11 = Symmetry in X</p> <p>12 = Symmetry in Y</p> <p>13 = Symmetry in Z</p> <p>14 = Axis (J around)</p> <p>15 = Axis (K around)</p> <p>16 = Axis (L around)</p> <p>17 = Symmetry with no reflection plane</p> <p>21 = 2-D condition in Y (apply to one face)</p> <p>(3 planes supplied at Y=-1,0,1)</p> <p>22 = Axisymmetric in Y, (apply to one face)</p> <p>(3 planes supplied, +/- 1 deg rotation about X)</p> <p>30 = Outflow (extrapolation)</p> <p>31 = Characteristic condition</p> <p>32 = Supersonic/subsonic inflow/outflow</p> <p>33 = Specified pressure outflow</p> <p>35 = Outflow (1st-order extrapolation)</p> <p>40 = Impose freestream</p> <p>41 = Nozzle inflow</p> <p>42 = Prescribed Q</p> <p>43 = Prescribed Q with slow start</p> <p>43 = Impose freestream</p> <p>44 = actuator disk</p> <p>45-46 = prescribed Q/inflow-outflow condition</p> <p>47 = Freestream/characteristic condition</p> <p>49 = Default (no change)</p> <p>51 = C-grid flow-through (specify on side) (J is C-direction)</p> <p>52 = C-grid flow-through (specify on side) (K is C-direction)</p>	None

	<p>52 = C-grid flow through (specify on side, L is C-direction)</p> <p>54 = Fold-over cut flow-through (fold-over in J)</p> <p>55 = Fold-over cut flow-through (fold-over in K)</p> <p>56 = Fold-over cut flow-through (fold-over in L)</p> <p>57 = C-grid at a wall (apply wall first) (J is C-direction)</p> <p>58 = C-grid at a wall (apply wall first) (K is C-direction)</p> <p>59 = C-grid at a wall (apply wall first) (L is C-direction)</p> <p>61 = Blank out region (set IBLANK=0)</p> <p>70 = Copy to</p> <p>71 = Copy from</p> <p>81 = Slotted wind-tunnel wall</p> <p>82 = Slotted wind-tunnel wall</p> <p>86 = Wind tunnel exit</p>	
IBDIR	<p>List of index directions:</p> <p>1 = positive J direction</p> <p>-1 = negative J direction</p> <p>2 = positive K direction</p> <p>-2 = negative K direction</p> <p>3 = positive L direction</p> <p>-3 = negative L direction</p>	None
JBCB	list of beginning j-indices	None
JBCE	list of beginning j-indices	None
KBCB	list of beginning k-indices	None
KBCE	list of beginning k-indices	None
LBCB	list of beginning l-indices	None
LBCE	list of beginning l-indices	None
XSYM	symmetry at X=0.0	1 if ibtyp=11 0 otherwise

YSYM	symmetry at Y=0.0	1 if ibtyp=12 0 otherwise
ZSYM	symmetry at Z=0.0	1 if ibtyp=13 0 otherwise
X2D	2-dimensional in X supply 3 planes in X=-1,0,1	0
Y2D	2-dimensional in Y supply 3 planes in Y=-1,0,1	1 if ibtyp=21 0 otherwise
Z2D	2-dimensional in Z supply 3 planes in Z=-1,0,1	0

\$HCUT

This input specifies a group of meshes that be used to create a hole cutting boundary, as well as which meshes will be cut by this hole cutting boundary. Entries in the \$BCINP namelist must be present for this method to work. Both solid boundary and symmetry plane information must be present in the \$BCINP namelist for the meshes that are used for the hole cutting boundary. If no \$HCUT namelist groups are specified and HCUT is equal to .TRUE. (in namelist group \$GLOBAL), a hole cutting boundary will be created from all solid boundaries in all of the \$BCINP namelists, and this hole cutting boundary will cut a hole in all meshes. There can be zero to many \$HCUT namelist groups specified in the user inputs.

Variable	Description	Default
NAME	Name of hole cutting group (must be any alphanumeric characters with no special symbols).	none
MEMBER	List of meshes that make up a group. This group of meshes must define a completely enclosed volume.	all meshes
INCLUDE	List of meshes that will have minimum holes cut by this group of meshes.	all meshes

EXCLUDE	List of meshes that will not have minimum holes cut by this group of meshes. This input is only used when INCLUDE is not specified, i.e., when all meshes have been specified.	none
OCTLEVEL	Integer level of octree refinements used to define automatic hole cutting boundaries.	Global value of OCTLEVEL
CNX,CNY,CNZ	Integer dimensions of Cartesian grids (hole maps) used for hole cutting.	Global values of CNX, CNY, CNZ
CARTX CARTY CARTZ	<p>Each of these inputs is a pair of numbers to set the min and max boundaries in the x, y, or z directions of the Cartesian boxes used in the automatic hole cutting. If they are left at their default values, the hole-cutting cartesian boundaries are set to extend just beyond the surfaces making up the hole cutter.</p> <p>This input can be used to limit the extent of the of the volume in which holes can be blanked out by the hole-cutters, and in some instances, can be used to help close off leaks in a hole-cutting surface.</p>	Global values of CARTX, CARTY, CARTZ

\$OUTER

If this input is not specified and OUTER is equal to .TRUE. (in namelist group \$GLOBAL) outer boundaries for all meshes will be specified automatically. This means that meshes which do not have boundary condition data given in the \$BCINP namelist will have outer boundary points specified on all mesh faces. The \$OUTER namelist group should be used only if a specific group of meshes needs to be included or excluded from automatic outer boundary specification. Only one \$OUTER namelist group

can be specified.

Variable	Description	Default
INCLUDE	List of meshes that will have outer boundaries determined from input boundary conditions.	all meshes
EXCLUDE	List of meshes that will not have the outer boundaries determined from input boundary conditions. This input is only used when INCLUDE is not specified, i.e., when all meshes have been specified.	none

\$BOUNDARY

Variable	Description	Default
CLOSED	Either .TRUE. or .FALSE. Set equal to .FALSE. when the surfaces do not describe a completely closed boundary.	.TRUE.
ISPARTOF	Mesh which contains boundary (must be any alpha-numeric characters with no special symbols). ISPARTOF is a string not exceeding ICHAR characters.	none
MHOLEIN	Meshes in which the named boundary causes holes. MHOLEIN's are specified as follows: MHOLEIN = 'mesh name 1', 'mesh name 2', etc. Each mesh name in MHOLEIN is a string not exceeding ICHAR characters.	none
NAME	Name of boundary (must be any alpha-numeric characters with no special symbols). A boundary name is a string not exceeding ICHAR characters.	none

\$SURFACE

Variable	Description	Default
ISPARTOF	Boundary to which surface belongs (must be any alpha-numeric characters with no special symbols). ISPARTOF is a string not exceeding ICHAR characters.	none
JRANGE, KRANGE, LRANGE	Ranges of indices that define surface. Composed of two values (a starting and an ending value) in each of the J, K, and L directions, respectively. Maximum allowable ranges are: JRANGE: 1,JMAX KRANGE: 1,KMAX LRANGE: 1,LMAX	Default: none
MESH NAME	Name of mesh that contains the surface (ISPARTOF) (must be any alpha-numeric characters with no special symbols). MESH NAME is used only for hole creation boundaries. Name is a string not exceeding ICHAR characters.	mesh name given in ISPARTOF for \$BOUNDARY.
NVOUT	Direction of normal outward from hole. NVOUT is required only for hole creation boundaries. Allowable values are "+J", "-J", "+K", "-K", "+L", and "-L".	none

\$BOX

Variable	Description	Default
ISPARTOF	Boundary to which box belongs (must be any alpha-numeric characters with no special symbols). ISPARTOF is a string not exceeding ICHAR characters.	none
XRANGE, YRANGE, ZRANGE	Ranges of X, Y, and Z coordinates that define box. Composed of two values (a minimum and a maximum value) in each of the X, Y, and Z directions, respectively.	none

\$REGION

Variable	Description	Default
ISPARTOF	Mesh which contains region (must be any alpha-numeric characters with no special symbols). ISPARTOF is a string not exceeding ICHAR characters.	none
NAME	Name of region (must be any alpha-numeric characters with no special symbols). A region name is a string not exceeding ICHAR characters.	none
TYPE	<p>Defines type of region being specified. If TYPE='HOLE', then hole with a fringe is created. If TYPE='INTR', then an interior region will be blanked out but with no fringe.</p> <p>If TYPE='UNBL', specified points will not be hole points (This input is used to change hole points back to field or interpolated boundary points).</p>	HOLE

\$VOLUME

Variable	Description	Default
ISPARTOF	Region to which volume belongs (must be any alpha-numeric characters with no special symbols). ISPARTOF is a string not exceeding ICHAR characters.	none
JRANGE, KRANGE, LRANGE	Ranges of indices that define volume. Composed of two values (a starting and an ending value) in each of the J, K, and L directions, respectively. Maximum allowable ranges are: JRANGE: 1,JMAX KRANGE: 1,KMAX LRANGE: 1,LMAX	none

\$LEVEL2

If this input is not specified and LEVEL2 is equal to .TRUE. (in namelist group \$GLOBAL) "level 2" interpolation for all meshes will be specified automatically. The \$LEVEL2 namelist group should be used only if a specific group of meshes needs to be included or excluded from "level 2" interpolation. Only one \$LEVEL2 namelist group (INCLUDE or EXCLUDE) can be specified.

Variable	Description	Default
INCLUDE	List of meshes that will have "level 2 interpolation" performed on them.	all meshes
EXCLUDE	List of meshes that will not have "level 2 interpolation" performed on them. This input is only used when INCLUDE is not specified, i.e., when all meshes have been specified.	none

3.3 Utility Codes

Several utility codes are supplied with PEGASUS in the /tools directory:

- **peg_setup**

- `peg_in`
- `peg_hole_surf`
- `p3d2peg`
- `make_WORK`
- `mesh_lister`
- `clean_holes`
- `peg_plot`
- `peg_diag`
- `peg_hole`
- `peg_proj`
- `peg_orph`
- `peg_mem`
- `XINtegrity`
- `XIN2sp`

`peg_setup` - This is a menu-driven script which gives the user options to initialize a PEGASUS problem, remove a PEGASUS problem, prepare to restart a PEGASUS run from the beginning, or redo the hole-cutting from the beginning. The first option requires that the user supply a multi-zone PLOT3D grid file containing all of the volume grids to be used. Also, you can supply an OVERFLOW input file to provide the boundary conditions. It executes `mesh_lister`, `peg_in`, `make_WORK`, `p3d2peg`, and `peg_hole_surf`. This option will split up the user-supplied multi-zone grid into individual meshes and place them in the `X_DIR` directory, ready for PEGASUS5. It creates a standard input file called `peg.in.raw` to help get you started. If an OVERFLOW input file is supplied, it will add the boundary condition data to the input file, and will also create a PLOT3D file of the solid surfaces of the configuration, called `hcutter.g`. This file makes it easy to plot and check that the solid-wall boundaries form a closed surface, which is necessary for the automatic hole cutting to work. See Section 4.2 for more information on the use of `peg_setup`.

The following six utility codes can be executed by running `peg_setup`, but can also be executed as stand-alone codes:

`peg_in [-no_add_te]` - Creates a minimal PEGASUS5 input file. It is written to a file called `peg.in.raw` with the intention that it be reviewed and edited if necessary before use. It requires the `mesh_list` file, which can be generated with the `mesh_lister` program. `peg_in` will also read the boundary condition data from an OVERFLOW input file and add this information into `peg.in.raw`. The mesh names must be present in the OVERFLOW input file, otherwise the boundary condition information can not be used by PEGASUS.

The `-no_add_te` flag will cause it to not add an extra solid surface (`ibtyp=-1`) at the trailing edge of a component where a flow-through boundary starts.

`peg_hole_surf` - creates an unformatted, multi-zone PLOT3D grid file, containing all of the surfaces used in automatic hole creation.

p3d2peg - breaks up the user-supplied PLOT3D file into single grid files which are placed in the /X_DIR directory.

make_WORK. -creates the WORK directory and its sub-directories which are needed within the project directory. If these are not created before-hand, PEGASUS5 will create them when it is executed.

mesh_lister - generates a list of mesh names for use by the other utility codes. The names will be generated from an OVERFLOW input file, if it exists; otherwise, the user will be prompted to either enter mesh names or assign default names to each grid in the user-supplied PLOT3D grid file.

clean_holes - removes the contents of /WORK sub-directories which are relevant to hole generation. Use this script if it is desired to re-run only the hole-generation processes again. If one one wants to re-run the entire PEGASUS process from the beginning, simply remove or rename the WORK directory.

The following utility codes are supplied for diagnostic purposes:

peg_plot [-sp] [-noorphfix] - produces an unformatted, multiple grid PLOT3D file (with IBLANKs) of the grid system. It reads the iblack and stencil data from the XINTOUT file, and so can only be used after PEGASUS5 has been run to completion (THRU = 'xintout'). Use **peg_hole** to plot the holes if you have only run with THRU = 'comp_hole'. The program prompts for the name of the output file, and for the level of fringes to plot, where the fringe level can be:

- 1 = single fringe: blank out level-2 and higher fringe points
- 2 = double fringe: blank out level-3 and higher fringe points
- 3 = all: only the minimum holes are blanked out

If the code was compiled using double-precision, one can produce a single-precision grid file by adding the command-line argument **-sp**.

It assigns an iblack value of 101 to first level ORPHAN points so they can be plotted; for example, use function 3 in PLOT3D. The code will automatically assign second-level ORPHAN points an iblack value of 1 (a valid interior point) unless used with the **-noorphfix** argument, in which case the iblack will be assigned a value of 101.

peg_diag - produces a diagnostic file for plotting. It can be produced after PEGASUS execution has been completed. The file that is created allows the user to visualize quality and cell difference parameter with PLOT3D or FAST (as well as any visualization software that reads PLOT3D grid and solution files). This file can be used with the grid file produced with **peg_plot**. There are no PHANTOM meshes included in this file. The values in this file are as follows:

Variable	Value or parameter	Grid point type
Q(1)	Quality Parameter 1.0	Legal Interpolated Boundary Point Field Point
Q(2)	Cell Difference Parameter 0.0	Legal Interpolated Boundary Point Field Point
Q(3)	Connecting Mesh Number Current Mesh Number 0.0	Legal Interpolated Boundary Point Field Point Orphan Point
Q(4)	1.0 0.0 -1.0	Field Point Legal Interpolated Boundary Point Orphan Point
Q(5)	Connecting Mesh Number 0.0 -1.0	Legal Interpolated Boundary Point Field Point Orphan Point

peg_hole - creates a grid file with IBLANK for the hole system that has been created. PEGASUS must be executed through (at least) the "comp_hole" process in order to use this utility. An unformatted, multiple grid, with IBLANK, PLOT3D file is produced. This code must be executed in the project directory.

peg_proj - creates diagnostic multi-zone PLOT3D grid and solution files, named `peg_proj.g` and `peg_proj.q`, which contain a zone for each solid-wall boundary surface. The grid file contains the grid-coordinates at the surface points, and the q file contains fields `l`, `dx`, `dy`, `dz`, and `l`. The `dx`, `dy`, and `dz` data are the components of the projection vector at each surface point for the projection of that point with the greatest magnitude. This data is best used by plotting velocity vectors at all the points, and coloring the vectors by velocity magnitude. Typically, it is best to scale the velocity vectors by a factor in the range 10 - 100. **peg_proj** creates a PLOT3D command file named `peg_proj.com`, which can be used to create this type of plot using PLOT3D. **peg_proj** also reports the maximum distance moved for each mesh, and onto which mesh that maximum projection occurred.

peg_orph - produces a formatted file with a list of the orphan points for all meshes or for a specified mesh. The output can also be directed to the screen.

peg_mem - estimates memory requirements for a given PEGASUS problem.

XINtegrity - checks the *XINTOUT* file produced by PEGASUS for internal consistency. This utility checks the accuracy of interpolations and determines whether interpolated boundary points are receiving information from the correct mesh elements. A PLOT3D file is produced with IBLANKS set equal to the negative of the mesh number from which interpolated points receive information. Orphan points are identified with IBLANK=2.

XIN2sp - converts a double precision *XINTOUT* file to single precision. This utility is used if it is desired to run the flow solver in single precision, but PEGASUS was run in double precision.

It is recommended that all of the utility codes be executed from the project directory, as most of them require information from the */WORK* and */X_DIR* directories.

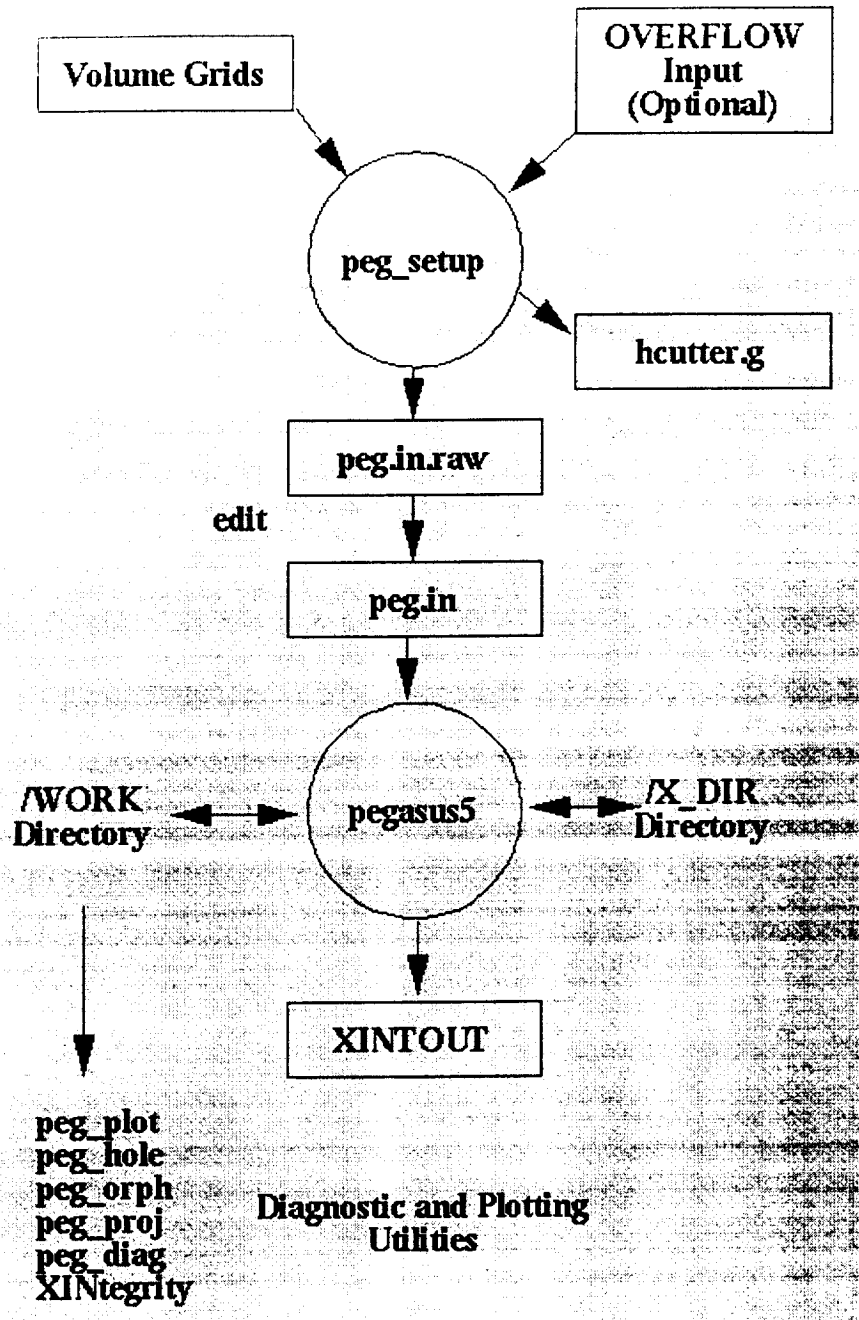


Figure 9. Data Flow Between Utility Codes and PEGASUS

3.4 Output File Formats

Standard Output

The standard output file contains the following:

- A summary of all user inputs.
- A final summary containing the following:
 - The total number of hole points
 - Number of interpolated boundary points for each mesh
 - Number of interpolation stencils for each mesh
 - Number of orphans in each mesh

XINOUT

The **XINTOUT** file contains all of the interpolation stencils for the fringe points and outer-boundary points, as well as an iblank array defining the hole-cuts. The file is unformatted, and contains the following four records for each mesh *M*.

Record Number	Variable Name
1	IBPNTS(M),IIPNTS(M),IIEPTR(M),IISPTR(M), MJMAX(M),MKMAX(M), MLMAX(M)
2	(JI(I),I=1,IINPTS(M)), (KI(I), I=1,IINPTS(M)), (LI(I), I=1,IINPTS(M)), (DXINT(I), I=1,IINPTS(M)), (DYINT(I), I=1,IINPTS(M)), (DZINT(I),I=1,IIPNTS(M))
3	(JB(I), I=1,IBNPTS(M)), (KB(I), I=1,IBNPTS(M)), (LB(I), I=1,IBNPTS(M)), (IBC(I),I=1,IBPNTS(M))
4	(IBLANK(I),I=I1,I2)

where $I1 = 1$ and $I2 = MJMAX(M)*MKMAX(M)*MLMAX(M)$

3.5 Directory Structure

PEGASUS 5.1 stores its files and information in a specific directory structure which it will create when it is first executed in a particular project directory. These directories can also be created in advance using the make_WORK script, which is also executed by the peg_setup script.

Execution of PEGASUS 5.1 must be performed in the project directory (see Figure 10). The user must supply the standard input file, see Section 3.2 for input descriptions) and the individual volume grids. The individual volume grids are stored in the X_DIR directory as unformatted, single-zone, PLOT3D grid files. These grid files are named "mesh_nameA.x", "mesh_nameB.x", etc, where mesh_nameA, mesh_nameB, ... correspond to the NAME variables in the SMESH namelists.

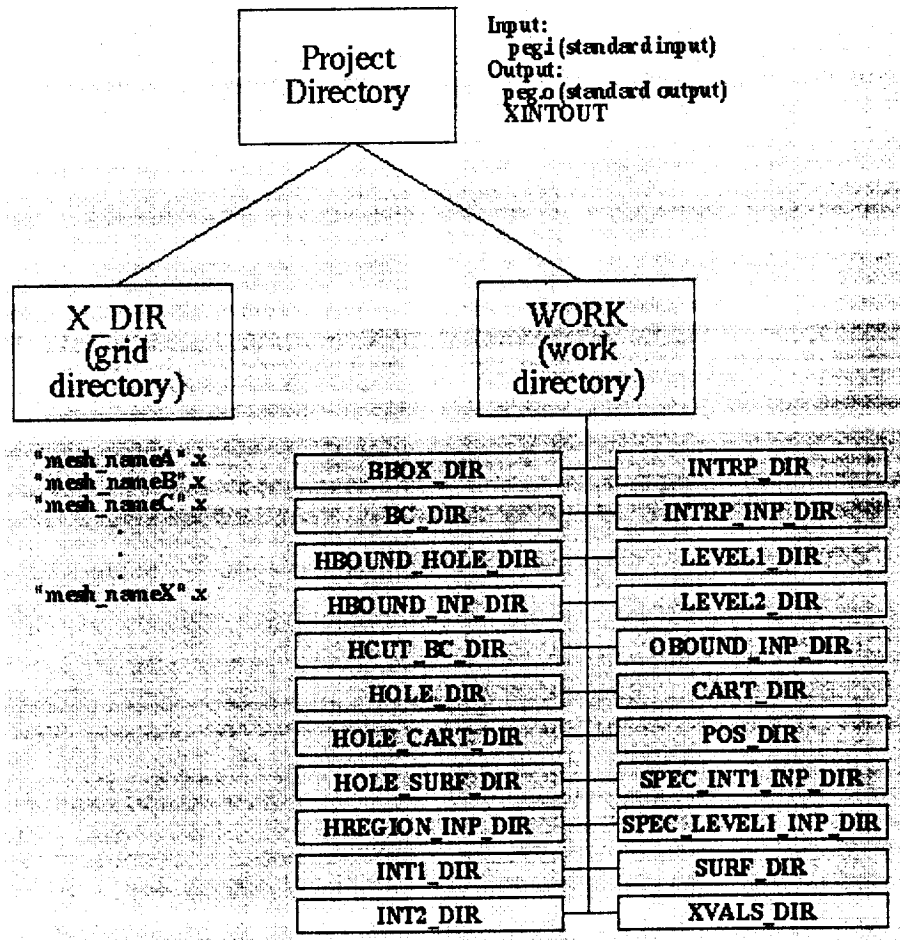


Figure 10. Directory Structure

PEGASUS 5 User's Guide

[Previous Section](#)

[Return to Table of Contents](#)

[Next Section](#)

octane13:rogers

stdin

Mon Aug 28 07:00:28 2000

hp11 / hp8apple

hp11 octane13:rogers Job: stdin Date: Mon Aug 28 07:00:28 2000

hp11 octane13:rogers Job: stdin Date: Mon Aug 28 07:00:28 2000

hp11 octane13:rogers Job: stdin Date: Mon Aug 28 07:00:28 2000

hp11 octane13:rogers Job: stdin Date: Mon Aug 28 07:00:28 2000

PEGASUS 5 User's Guide		
Previous Section	Return to Table of Contents	Next Section

Section 4: USING PEGASUS

4.1 Setting Up the Input

The following are steps that should be taken when starting a new PEGASUS problem.

1. Create a project directory.
2. Prepare the volume grids as a multi-zone, unformatted, PLOT3D grid file.
3. Optionally, prepare an OVERFLOW (Refs. 3-5) input file containing the names and the boundary conditions for each mesh.
4. Run the `peg_setup` script. Choose option 1, which helps prepare all files necessary to run **PEGASUS 5**. You will be asked for the names of the grid and, if available, the OVERFLOW input file.
5. Exit **peg_setup**. This will have split the volume grid file into individual volume grid files in a subdirectory named **X_DIR**, using the `p3d2peg` utility. This will also have generated a file named *peg.in.raw*.
6. If an OVERFLOW input file was supplied, the *peg.in.raw* will contain boundary condition data in the BCINP namelist. If an OVERFLOW input file was not supplied, this boundary condition information must be added to the input file.
7. If the boundary condition information was added by hand, run the utility `peg_hole_surf` to generate the file *hcutter.g*. If an OVERFLOW input file was supplied, the **peg_setup** script will have already performed this step.
8. The *hcutter.g* file is a PLOT3D grid file containing the solid surfaces of the configuration. Using a plotting package, such as PLOT3D, carefully examine *hcutter.g* for gaps in the geometry that may affect hole generation (See Section 4.4, Operational Hints). Modify the *peg.in.raw* to close any gaps in the geometry by adding additional surfaces to the list of boundary conditions in the \$BCINP namelists, and use IBTYP=-1 for these additional surfaces. IBTYP=-1 denotes a solid surface which will be used by the automatic hole cutting, but which will not be excluded from the list of outer boundary surface points requiring interpolation data.
9. Once corrections have been made, rename *peg.in.raw* to *peg.in*.

4.2 Running PEGASUS

1. Run PEGASUS with the command:

pegasus5 < peg.in > peg.out

2. When PEGASUS execution is completed, the *XINTOUT* file will be generated, along with the PEGASUS log file. The *peg.out* file will contain a table that lists, for each mesh, the number of interpolated points, stencils, and orphans. The log file contains information about which processes were executed, and the amount of execution time required. If PEGASUS was compiled in double precision, the *XINTOUT* file will be double precision. A single precision *XINTOUT* file can be generated by running the XIN2sp utility code.
3. Generate a composite grid file with the *peg_plot* utility.
4. The restarting capability is a very powerful feature of PEGASUS 5.1. The restart function is modeled after the UNIX "make" utility. A new mesh or input, or a changed mesh or input will cause PEGASUS to perform only the processes that are affected by these modifications. For example, if a new mesh is to be added to the current configuration, the mesh is simply placed in the */X_DIR* directory and the mesh name added to the *peg.in* file. When PEGASUS is executed, the appropriate processes are performed to account for the modifications that were made. This also works for changes or additions in the input file *peg.in*.

The restart capability allows for an incremental buildup of the PEGASUS solution. Generally, it is effective to initially run PEGASUS with a minimal input file, possibly simply the *peg.in.raw* file generated by *peg_setup*. The first run of PEGASUS will do all of the necessary interpolations. The interpolation step should never have to be done again, unless a mesh is changed or added. Subsequent runs of PEGASUS can deal with other issues, such as manipulating the hole-cutting.

This restart capability is also very useful when PEGASUS abnormally terminates due to a system crash or when resource limits (e.g., CPU limits) are exceeded. Since PEGASUS retains all results in its work directory for all processes that have been completed, when PEGASUS is restarted, execution begins with the process that was executing during the termination. The user in this case only needs to re-execute PEGASUS; no changes in inputs are required.

4.3 Using Externally Generated Hole Definitions

PEGASUS uses a general purpose hole-cutting methodology that will cut good holes in the majority of cases for unmodified Chimera grid systems. However, there are situations where a user might want to use another hole-generation methodology. Or, a user might have legacy grid systems with previously generated hole definitions which have been successfully applied in the past. In those cases, it may not be necessary or desirable to use the hole-cutting methodology in PEGASUS; however, a user may still want to take advantage of other aspects of PEGASUS, such as level 2 interpolation. To accommodate these situations, a methodology has been developed whereby these externally defined hole definitions can be used within PEGASUS 5.1.

To use external hole definitions, a PLOT3D file with IBLANK records must be provided, with IBLANK=0 specified for each hole point. The IBLANK records may be generated by any means at the user's disposal.

- Run the utility code *p3d2peg*. The PLOT3D grid file and the *mesh.list* file (generated by the utility

code `mesh_lister`) must be in the project directory. `p3d2peg` will ask if you want to use the existing IBLANK file for hole definition (answer in the affirmative). If `p3d2peg` detects grid files in the `/X_DIR` directory, it will ask if you want to replace them. Unless one or more grids have been modified since the last PEGASUS run, answer "no". The IBLANK records will be copied into the appropriate `/WORK` subdirectories.

- Edit the `peg.in` file to include "HCUT=.FALSE.," in the \$GLOBAL input record.
- Run PEGASUS in the normal manner. If this is the first PEGASUS run, the entire interpolation process will be executed. If the interpolation process has been done earlier, the restart capability of PEGASUS will use the existing interpolation information to generate an `XINTOUT` file with the externally generated hole definitions.

4.4 Operational Hints

4.4.1 Gaps in Surfaces

PEGASUS 5.1 is highly automated, in the sense that much less user input is required to obtain a chimera interpolation file than was required in earlier versions of PEGASUS. However, automation generally requires making some assumptions about the input; if these assumptions are incorrect, PEGASUS may do something other than what the user expects or desires, resulting in varying degrees of user distress. One assumption is that hole cutting surfaces are watertight, in the sense that no gaps occur in the aggregate of the solid walls of the aerodynamic configuration. However, some common approaches to defining solid wall boundary conditions in flow solvers may, to PEGASUS, represent gaps which will adversely affect the generation of the Cartesian meshes that PEGASUS uses for cutting holes.

Consider, for example, the solid surface generated when the OVERFLOW input used in the 3-element airfoil example is used to generate the `peg.in` file using the `peg_in` utility code. Figure 11 depicts the region near the corner under the back of the main wing.

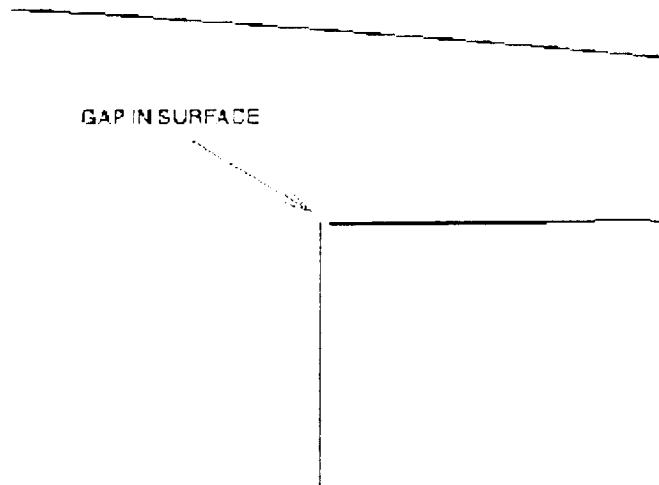


Figure 11. Surface Gaps

In an effort to not specify the corner point twice as a solid boundary condition, the corner point is included in the specification of the vertical surface, but not in the horizontal surface. This is fine as far as the flow solver is concerned, but PEGASUS sees this as a gap in the surface.

Figure 12 depicts the trailing edge of the wing. The upper and lower surfaces of the trailing edge are from two different meshes; these meshes overlap one grid point past the end of the trailing edge. Solid surface boundary conditions are specified up to the end of the trailing edge; beyond that point, the boundaries are updated via chimera interpolation. As in the corner example, PEGASUS sees a gap in the surface.

The effects of these gaps become apparent during the hole-cutting process. Recall from Section 2.1 that the outside region is defined by marching inward from the corners of the Cartesian mesh; any unidentified Cartesian element that is adjacent to an outside element must also be an outside element. It is clear that if the Cartesian mesh is fine enough, outside elements can "leak" through the gaps in the surface and misidentify elements inside the body as outside elements. Therefore, no hole will be cut.

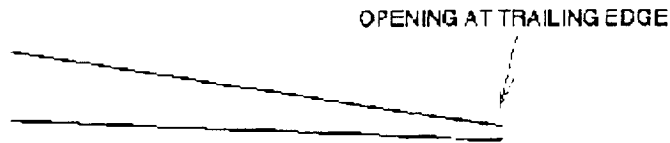


Figure 12. Trailing Edge Opening

The remedy for gaps in the surfaces depends on the nature of the openings. In the case of the corner gap, the *peg.in* file can simply be modified to extend the solid boundary of the horizontal surface to the corner. The trailing edge gap, however, requires the addition of another boundary surface in the \$BCINP namelist, and use IBTYP=-1 for this additional surface. IBTYP=-1 denotes a solid surface which will be used by the automatic hole cutting, but which will not be excluded from the list of outer boundary surface points requiring interpolation data.

There is always a possibility that a small gap, such as that shown in the airfoil trailing edge in Figure 12. If the gap is smaller than the vertical spacing of the Cartesian mesh, the surface is closed as far as the Cartesian mesh is concerned. In some cases, the gap could be up to twice the corresponding spacing of the Cartesian mesh and still result in a good hole definition.

Other methods for fixing surfaces with gaps is with the use of auxiliary (i.e., "phantom") meshes. These meshes are designated in the standard input file in the \$MESH namelist by setting PHANTOM=.TRUE. The solid surfaces of a PHANTOM mesh are to be designated in the BCINP namelist for the phantom mesh. The phantom meshes do not participate in the interpolation, and do not appear in the output *XINTOUT* file.

Finally, problems with automatic minimum hole cutting can be remedied with the use of the manual hole cuts, which use the hole cutting algorithm in PEGSUS 4.1. Manual hole cutting surfaces are specified using the \$BOUNDARY and \$SURFACE namelists.

4.4.2 Thin Trailing Edges

The hole cutting algorithm is a very general approach, and can accommodate unmodified chimera grids systems with overlapped collar grids and multiple interior regions (which occur, for example, in the 3-element airfoil example). However, modern chimera systems often have a very large size range of surface features that the Cartesian hole cutting meshes must resolve. In the case of very fine features on a large configuration, the resolution required may be excessive. Consider, for example, the trailing edge of slat in the 3-element-airfoil example. Figure 13 depicts a point inside the

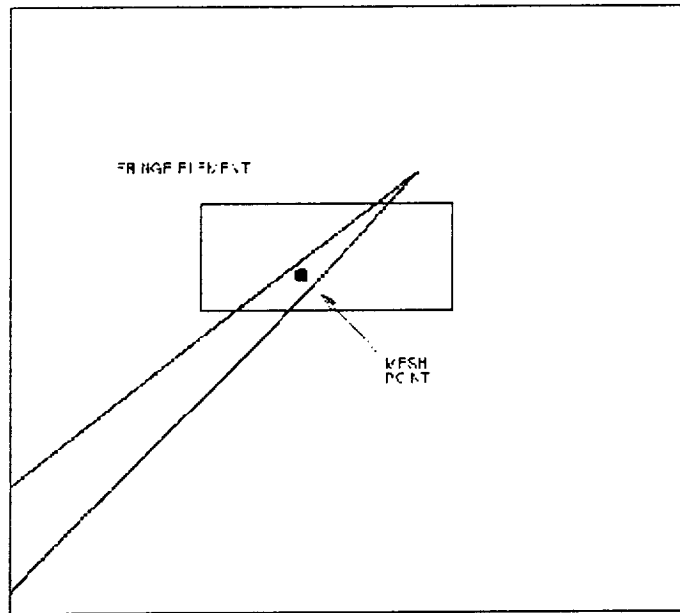


Figure 13. Hole Point Identification with Thin Trailing Edge

trailing edge of the slat. The solid surfaces of the trailing edge are encompassed by fringe elements, only one of which is shown for purposes of clarity. Recall from Section 2.1 that points inside fringe elements use a "line-of-sight" algorithm to determine their status. To reiterate, if a clear line-of-sight exists from a point in a fringe element to another non-fringe element, then the point adopts the status of the non-fringe element (either "outside" or "inside"). However, the point in this example may not have a clear "line-of-sight" to a non-fringe element, and therefore cannot be identified in this manner. If a point cannot be identified, it is assumed to be an outside (i.e., field) point. If the field point is next to a hole point, it will become a candidate for interpolation; being inside the slat, there will be no viable interpolation stencil, and the point will be an orphan.

Fortunately, points in thin trailing edges can often be blanked as desired using the `OFFSET` input in the `$GLOBAL` or `$MESH` input records. Setting `OFFSET = 2`, for example, grows hole regions by blanking any non-hole point within 2 indices of a hole point. In this manner points in thin trailing edges that are destined to be orphaned may be correctly blanked.

Also, multiple hole cutting groups can be used in PEGASUS. Separate hole cutters can be designated, for example, for the slat, wing, and flap separately. The Cartesian cutters are automatically sized

according to the solid surfaces in the group. Specifying a separate hole cutter around the slit will automatically increase the default resolution of the Cartesian hole-cutting mesh in the region of the trailing edge. An example of input which invokes multiple hole cutters is reproduced in Appendix B. Finally, the default logical dimensions of the Cartesian hole cutter may be increased until there is sufficient mesh resolution to resolve the trailing edge.

4.4.3 Projection

The value for `DISTANCE` in the `SGLOBAL` and `$MESH` namelists will control how many points can be projected. If `DISTANCE` is too large, it is possible for PEGASUS to project points too far, and actually move them from one component to another. It is advisable to routinely check the results of the projection by checking the maximum distance projected. This is reported in the log file, and is also easy to check by running the utility `peg_proj`.

PEGASUS 5 User's Guide		
Previous Section	Return to Table of Contents	Next Section

octane13:rogers

stdin

Mon Aug 28 07:01:33 2000

hp11 / hp8apple

hp11 octane13:rogers Job: stdin Date: Mon Aug 28 07:01:33 2000

hp11 octane13:rogers Job: stdin Date: Mon Aug 28 07:01:33 2000

hp11 octane13:rogers Job: stdin Date: Mon Aug 28 07:01:33 2000

hp11 octane13:rogers Job: stdin Date: Mon Aug 28 07:01:33 2000

PEGASUS 5 User's Guide		
Previous Section	Return to Table of Contents	Next Section

Section 5: REFERENCES

1. Suhs, N.E. and Tramel, R.W., "PEGASUS 4.0 User's Manual", AEDC-TR-91-8, November 1991.
2. Aftosmis, M. J., "Solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries", <http://george.arc.nasa.gov/~aftosmis/vki/vki97.html>1997.
3. Buning, P. G., Jespersen, D. C., Pulliam, T. H., Chan, W. M., Slotnick, J. P., Krist, S. E., Renze, K. J., "OVERFLOW User's Manual, Version 1.8," NASA Langley Research Center, Hampton VA, 1998.
4. Jespersen, D. C., Pulliam, T. H., and Buning, P. G., "Recent Enhancements to OVERFLOW," AIAA Paper 97-0644, Jan. 1997.
5. Jespersen, D. C., "Parallelism and OVERFLOW," NAS Technical Report NAS-98-013, October 1998. <http://www.nas.nasa.gov/Pubs/TechReports/NASreports/NAS-98-013/>
6. Rogers, S. E., Kwak, D., and Kiris, C., "Numerical Solution of the Incompressible Navier-Stokes Equations for Steady-State and Time-Dependent Problems," AIAA Paper 89-0463, January 1989. Also in Proceedings of the Tenth Australasian Fluid Mechanics Conference, Melbourne, Australia, Dec. 1989. Published in *AIAA Journal.*, Vol. 29, No. 4, April 1991, pp. 603--610.
7. Chan, W. M., Buning, P. G., and Krist, S. E., "PROGRD User's Manual, Version 0.5," NASA Ames Research Center, Moffett Field, CA, Aug. 1998.

PEGASUS 5 User's Guide		
Previous Section	Return to Table of Contents	Next Section

PEGASUS 5 User's Guide

[Previous Section](#)

[Return to Table of Contents](#)

Section 6: APPENDICES

APPENDIX A: Glossary

Term	Definition
Donor Mesh	Mesh that provides information by interpolation to another mesh.
Field Point	A point within computational domain (IBLANK=1) that is updated by the flow solver or boundary conditions.
Fringe Point	A hole boundary point.
Hole	Collection of hole points, i.e., points in a mesh that have been blanked.
Hole Boundary	Collection of hole boundary points (fringe points) surrounding a hole.
Hole Boundary Point	Point on a hole boundary, by definition a point between a hole point and field point.
Hole Map	A cartesian grid that has been developed to cut minimum holes.
Hole Point	A mesh point that has been specified to be within a hole. This point is considered to be out of the computational domain (IBLANK=0).

Example of PEGASUS Input File (*peg.in*)

(3-Element Airfoil)

EXAMPLE 1 - Automatic Hole Cutting and Outer Boundary Specification:

The following input file relies on the automatic definition of the minimum hole cutter, which uses all of the solid walls to cut holes out of all of the zones.

```
$GLOBAL
PROJECT = .F.,
LEVEL2 = .T.,
FRINGE = 2.,
CNY = 3.,
OFFSET = 2.,
$END
$MESH
NAME = 'slat',
$END
$MESH
NAME = 'main',
$END
$MESH
NAME = 'flap1',
$END
$MESH
NAME = 'flapte1',
$END
$MESH
NAME = 'wake',
$END
$BCINP
ISPARTOF = 'slat',
IBTYP = 5, 51, 21,
IBDIR = 2, 2, 3,
JBCS = 15, 2, 1,
JBCE = 107, 14, -1,
KBCS = 1, 1, 1,
KBCE = 1, 1, -1,
LBCS = 1, 2, 1,
LBCE = -1, -2, 1,
$END
$BCINP
ISPARTOF = 'main',
IBTYP = 5, 47, 47, 21, 47, -1,
IBDIR = 2, 1, -1, 3, -2, 2,
JBCS = 121, 1, 321, 1, 1, 231,
JBCE = 231, 1, 321, -1, -1, 232,
KBCS = 1, 1, 1, 1, -1, 1,
KBCE = 1, 101, 101, -1, -1, 1,
LBCS = 1, 2, 2, 1, 2, 1,
LBCE = -1, -2, -2, 1, -2, -1,
```

```

$SEND
$BCINP
ISPARTOF = 'flap1',
IBTYP = 5, 21,
IBDIR = 2, 3,
JBCS = 21, 1,
JBCE = 121, -1,
KBCS = 1, 1,
KBCE = 1, -1,
LBCS = 1, 1,
LBCE = -1, 1,
$SEND
$BCINP
ISPARTOF = 'flap1te',
IBTYP = 5, 21,
IBDIR = 1, 3,
JBCS = 1, 1,
JBCE = 1, -1,
KBCS = 4, 1,
KBCE = 28, -1,
LBCS = 1, 1,
LBCE = -1, 1,
$SEND
$BCINP
ISPARTOF = 'wake',
IBTYP = 5, 5, 47, 21, -1,
IBDIR = 1, -2, -1, 3, -2,
JBCS = 1, 1, 151, 1, 61,
JBCE = 1, 61, 151, -1, 62,
KBCS = 1, 61, 1, 1, -1,
KBCE = 61, 61, 61, -1, -1,
LBCS = 1, 1, 2, 1, 1,
LBCE = -1, -1, -2, 1, -1,
$SEND

```

EXAMPLE 2 - Automatic Hole Cutting Using Three Separate Hole Cutters:

By adding these \$HCUT namelist inputs to the input file in the previous example. These namelists define three hole-cutting surfaces, one for the slat, one for the main wing, and one for the flap.

```

$HCUT NAME = 'slathole',
MEMBER = 'slat',
INCLUDE = 'main',
$SEND
$HCUT NAME = 'winghole',
MEMBER = 'main', 'wake',
INCLUDE = 'slat', 'flap1',
$SEND
$HCUT NAME = 'flap1hole',
MEMBER = 'flap1', 'flap1te',
INCLUDE = 'main', 'wake',
$SEND

```


APPENDIX C

Obsolete Inputs

The following inputs were available in versions of PEGSUS 4.1 but are no longer accepted as inputs in PEGASUS 5.1.

NAMELIST	Obsolete Input
\$GLOBAL	QUALITY
\$MESH	LINK, ADDLINK, ADDHOLE, QUALITY

PEGASUS 5 User's Guide	
Previous Section	Return to Table of Contents