Pegasus5: An Automated Pre-Processor For Overset-Grid CFD

Stuart E. Rogers

Computational Aerosciences Branch
NASA Supercomputing Division
NASA Ames Research Center, Moffett Field, CA
stuart.e.rogers@nasa.gov

13th Symposium on Overset Composite Grids
October 17th, 2016, Mukilteo, WA

Acknowledgments

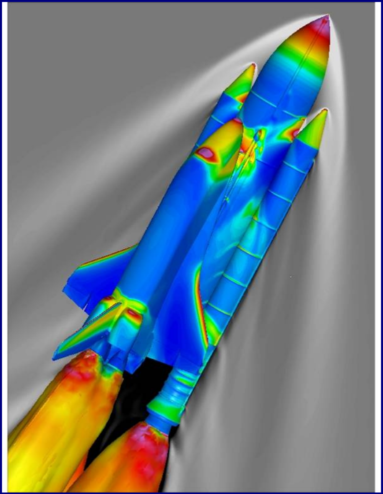
- Pegasus5 primary authors:
  - Norman Suhs
  - William Dietz
  - Stuart Rogers
- Developed with funding from:
  - NASA/Boeing/McDonnell-Douglas Advanced Subsonics Transport Program
  - NASA Information Power-Grid Program
  - NASA Space Shuttle Program
  - NASA Orion/MPCV Program

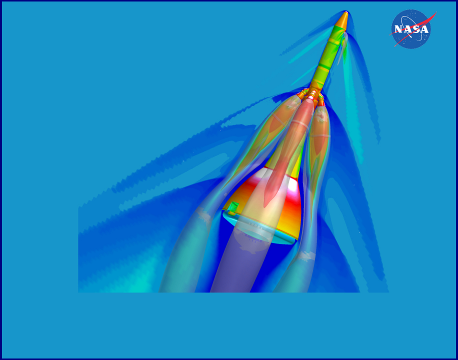
- Pegasus5 is co-winner of the 2016 NASA Software of the Year Award

Outline

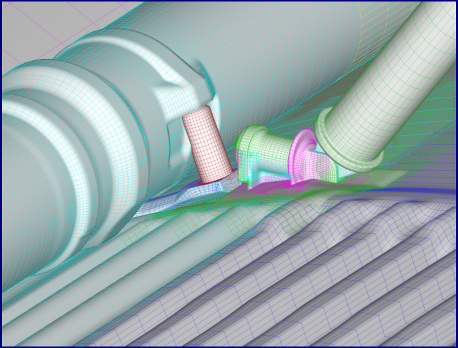
- Introduction and background
- Understanding overset-grid work flow
- Nomenclature
- Pegasus5 features and automation
- Overview of usage
    - Required input
    - Basic usage
    - Understanding the output
    - Advanced usage and overcoming problems

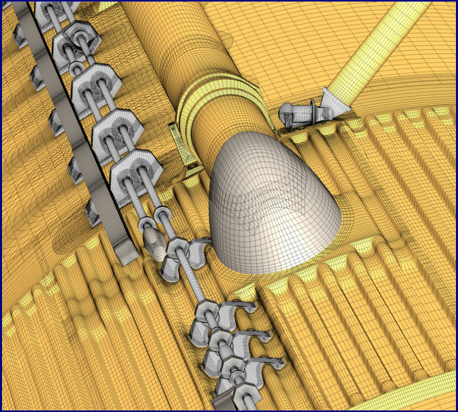
The Oversetting Challenge





The Oversetting Challenge





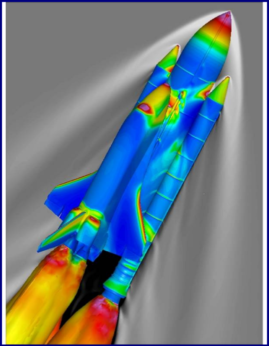
Background

What is Computational Fluid Dynamics (CFD)?

- Solving mathematical equations governing fluid flow
- Used extensively in all manner of aerodynamic analysis

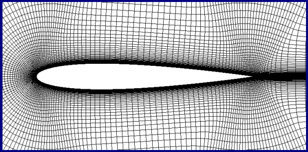
Why does the Aerospace Industry Need CFD?

- Cost effective versus wind-tunnel
- Simulate actual flight conditions
- Run many simulations covering entire flight envelope
- In-depth investigation of single simulation



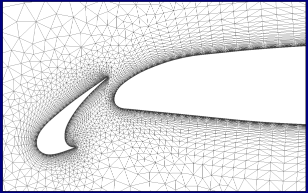
What Are Structured Grids?

- Ordered connection of blocks or cells
- Requires less computer time and memory
- Grid generation can be very difficult



What Are Unstructured Grids?

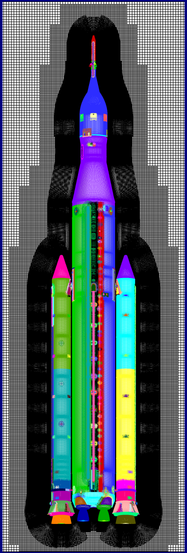
- Randomly connected polyhedra
- Much easier grid generation
- Require 2x to 10x more computer time and memory



Background: Overflow CFD Flow Solver

What is Overflow?

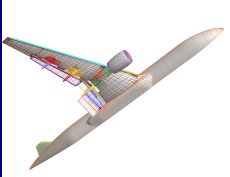
- NASA developed structured CFD solver
- One of the most extensively used CFD solvers in NASA
- Used by every major NASA vehicle program
- Most users of Pegasus5 use the Overflow solver

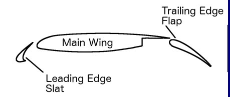


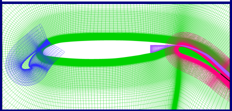
Background: Overset Structured Grids

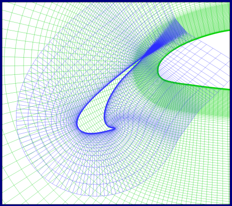
What are Overset Structured Grids?

- Split complex geometry problems into simple components
- Enables highest quality grids in viscous boundary layer
- Requires software capable of integrating the overlapping grids
- An entire aircraft requires 100s of millions of grid cells

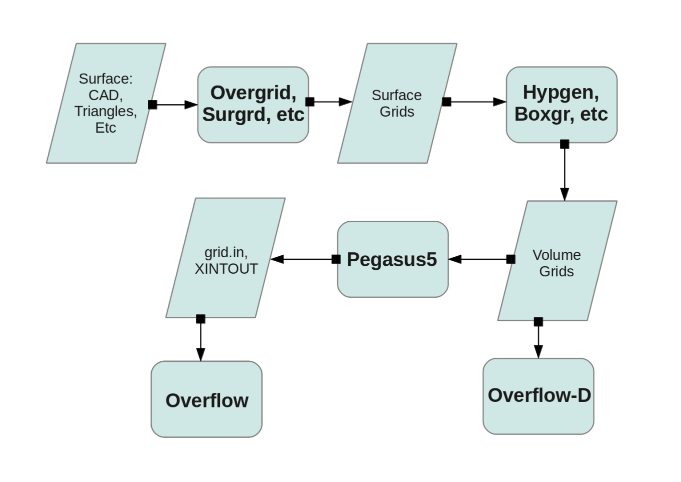








Overset-Grid Workflow For Overflow



Nomenclature

- **Grid System:** A collection of zones together with boundary conditions and connectivity data ready to input into a flow solver
- **Zone or Mesh:** a single structured grid composed of ordered points
- **Cell:** a hexahedron composed of 8 grid points and 6 faces
- **Grid point:** a singe point in a zone identified uniquely by its j,k,l indices
- **Fringe point:** a grid point which will be updated in the flow solver via interpolation of the solution from a neighboring zone
- **Outer-boundary fringe point:** a fringe point on the boundary edge of a zone
- **Hole-boundary fringe point:** a fringe point adjacent to a hole point

Nomenclature, continued

- **Hole point:** a grid point which has been "blanked out" and whose data will not be used by the flow solver
- **Orphan point:** a fringe point for which a valid donor cell cannot be found
- **Interior point:** a grid point which does not lie on the zonal boundary
- **Iblank value:** each grid point is assigned an integer value to denote the type of point:
  - iblank<0: fringe point
  - iblank=0: hole point
  - iblank=1: active interior point
  - iblank=101: orphan point

Pegasus5 Goals and Features

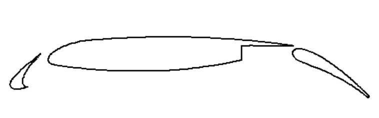
- Fifth-generation overset software
  - Written in 1998-2000 as a replacement for *PEGSUS4*
- Primary goal: automation of most of the oversetting process
  - Complexity of CFD problems continues to grow
  - Hundreds of zones, hundreds of millions of grid points
  - Manual control of process is intractable
- Requires all-new approach to:
  - Input requirements
  - Hole-cutting
  - Overlap optimization
- Requires ease-of-use improvements:
  - Parallelization
  - Restarting
  - Projection
- Maintained *PEGSUS4* manual hole-cutting capabilities
- Pegasus5 is dramatically easier to use than previous versions, but still requires knowledgeable user

Pegasus5 Approach

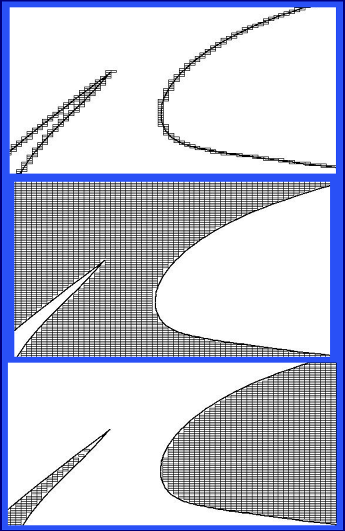
- Use an Overflow-like namelist input file
  - Boundary conditions provide most of the required input about geometry and grid topology
- Use Fortran90
  - Extensive use of defined-type data and modules
  - Extensive use of process templates and data templates
- Oversetting task broken into discrete processes
  - Input to each process saved as one or more disk files
  - Output from each process saved as one or more disk files
  - Facilitates parallelization
  - Enables restarts from partial or aborted runs
  - Enables rapid restarts after changes to input
- Uses lots of CPU time and disk I/O

Auto-Hole Cutting Using a Cartesian Map

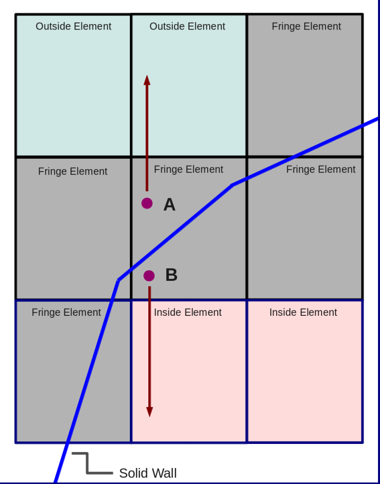
2D Multi-Element Airfoil Example



- Identify solid-wall surfaces and overlay with Cartesian map
- **Fringe elements:** Cartesian elements which intersect walls
- **Outside elements:** Identify with painting algorithm
- **Inside elements:** All other elements



- Loop through all volume grid points
- Points outside Cartesian map marked as outside
- Points in the **Outside Elements** marked as outside
- Points in the **Inside Elements** are blanked
- Points in the **Fringe Elements** use line-of-sight test:
  - Point A has clear line-of-sight to an Outside Element: outside point
  - Point B has clear line-of-sight to an Inside Element: blank point



Interpolation Stencil Search

- Pegasus5 searches for all possible interpolation stencil donors from all zones **for every single grid point**
- Alternating Digital Tree (ADT) is created for all zones
- For a given grid point and a given donor zone, an ADT lookup provides a near-by cell, then a stencil-jumping approach finds the exact donor cell and interpolation stencil
- All possible donors cells are stored for every grid point

Fringe Point Identification

- A fringe point is one which requires updating in the flow-solver via interpolation from a neighboring zone
- Outer-boundary fringe points:
  - All points on the boundary of a zone that do not receive a flow-solver boundary condition are identified as outer-boundary fringe points
- Hole-boundary fringe points:
  - Points adjacent to a hole point are identified as hole-boundary fringe points
- Single, double, or triple layers of fringe points can be requested

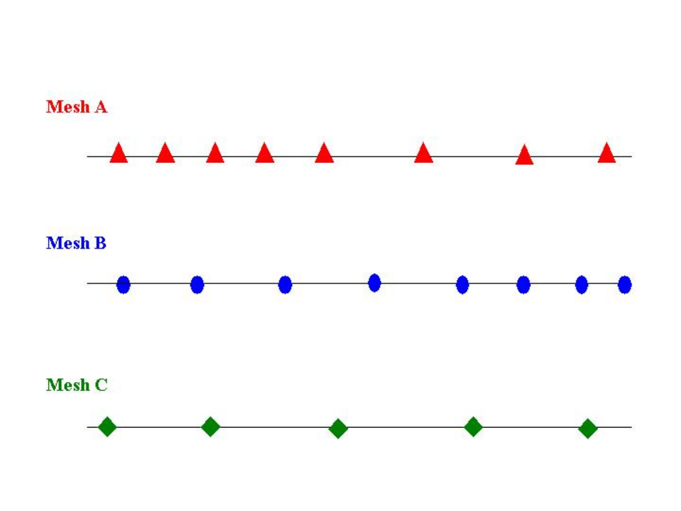
Level-1 Interpolation

- For each fringe point, the best possible interpolation stencil is chosen amongst all valid donor cells
- When multiple donors are available, selection is based on a measure of interpolation quality and relative cell size
- Any fringe point which does not have a valid donor is denoted as an orphan point

Level-2 Interpolation

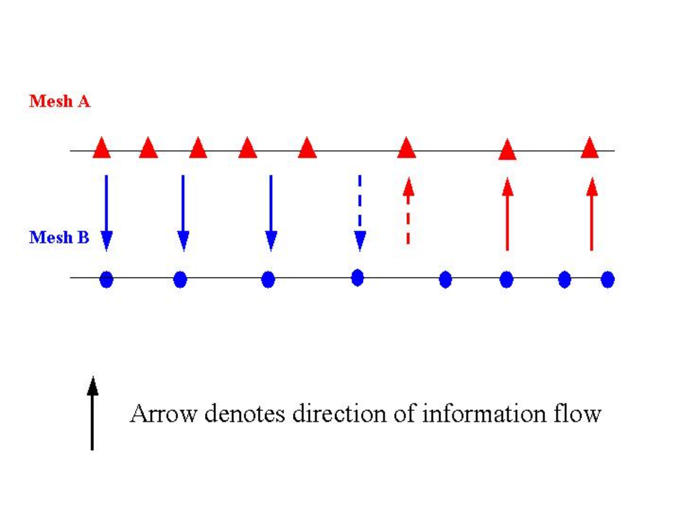
- Optimization of overlap between zones
- Interpolation points added after Level-1 interpolation
- Has effect of expanding the automatically-cut holes and shrinking the outer edges of overlapping zones
- Finest grid points remain active interior points
- Coarser grid points are interpolated from available donor cells of finer neighboring zones
- Methodology is robust, requires no user inputs, and maximizes communication between overlapping zones

One-Dimensional Example



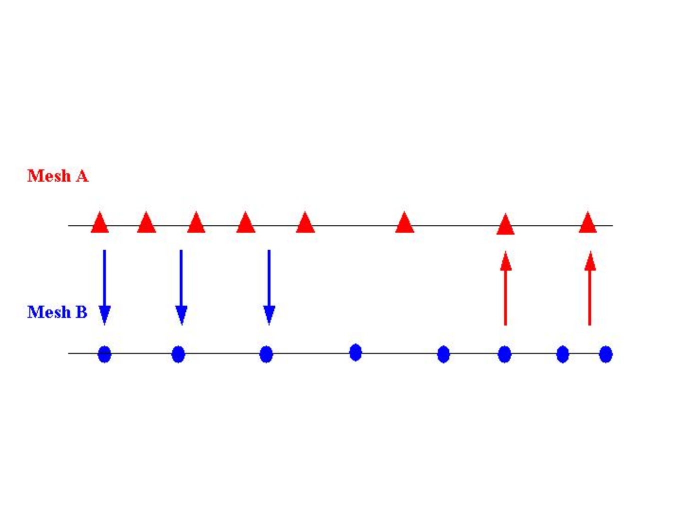
Level-2 Interpolation

Step 1: Interpolate Between Meshes



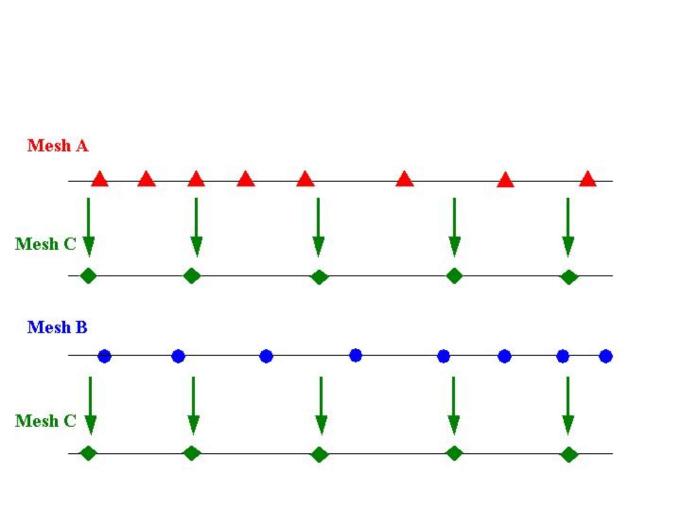
Level-2 Interpolation

Step 2: Remove Invalid Interpolations



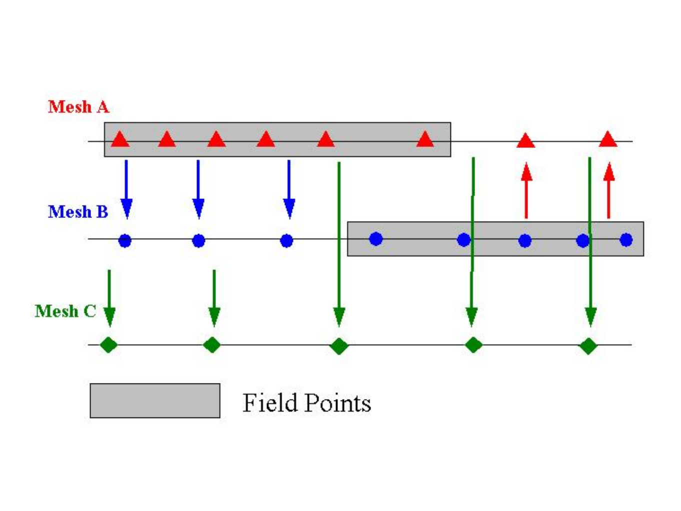
Level-2 Interpolation

Step 3: Repeat For Other Meshes



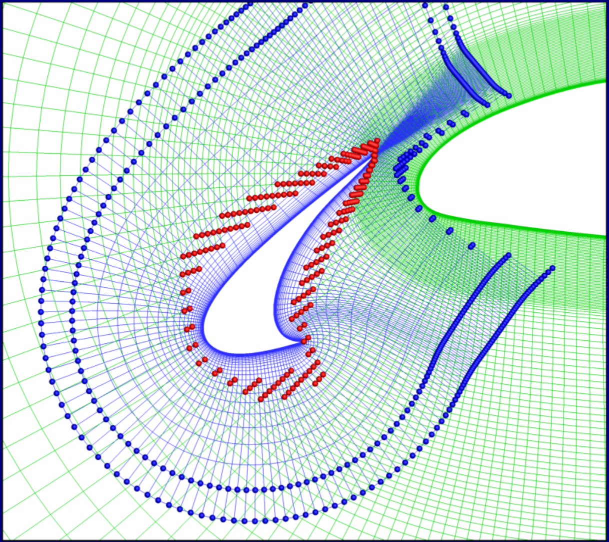
Level-2 Interpolation

Step 4: Keep Finest Mesh Points



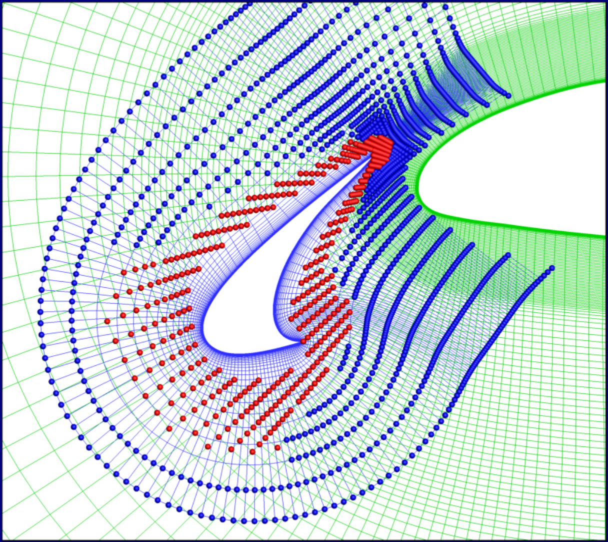
Optimized Overlap Example: Multi-Element Airfoil

Level 1 Fringes



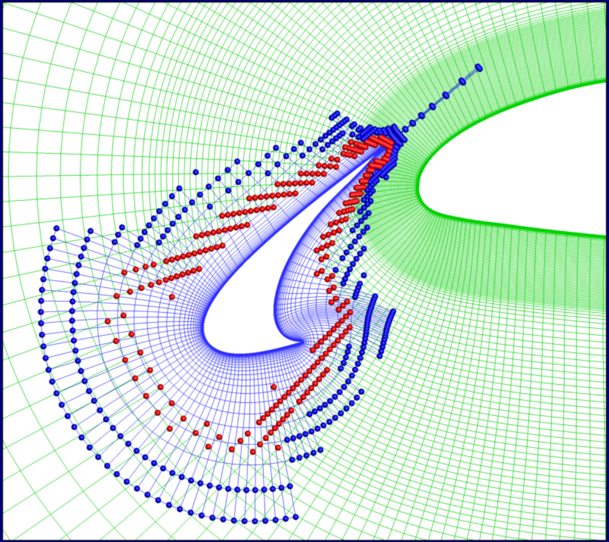
Optimized Overlap Example: Multi-Element Airfoil

Level 2 Fringes

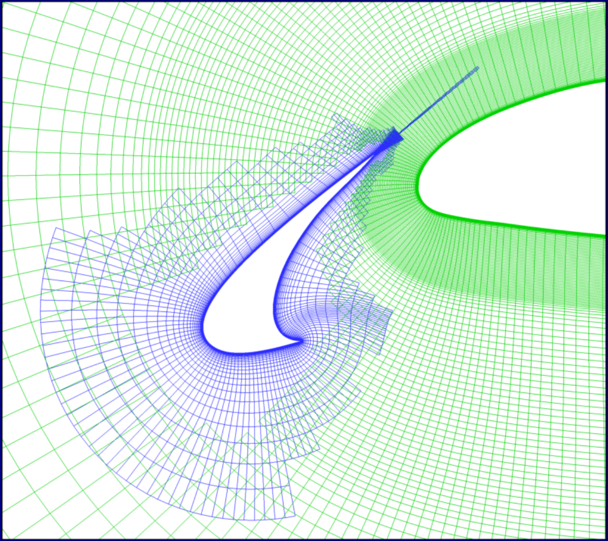


Optimized Overlap Example: Multi-Element Airfoil

Virtual Fringes



Optimized Overlap Example: Multi-Element Airfoil

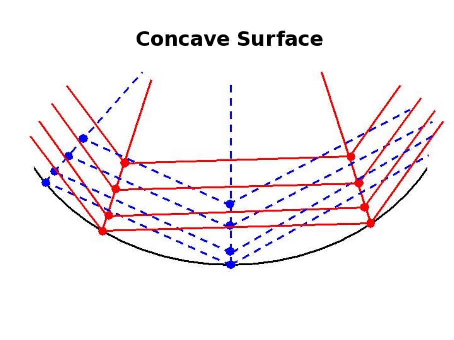


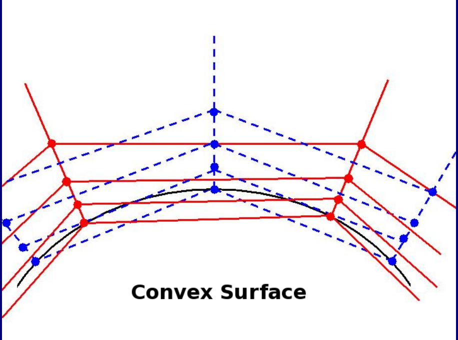
Projection

- Corrects interpolation problems that may occur on overlapping curved viscous surfaces
- Cell-aspect ratio typically > 1000 near viscous walls
- Pegasus5 projection step alters interpolation coefficients, not actual grid points
- Projection is performed internally and typically requires no user input

Problem:

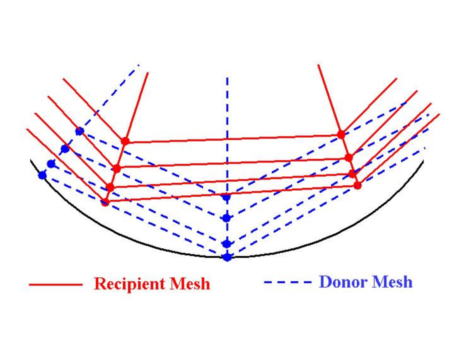
Linear Discretization on Curved Surfaces

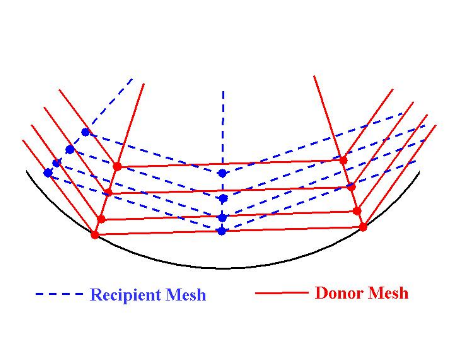




Solution: Projection

Points are Projected for Interpolation Only





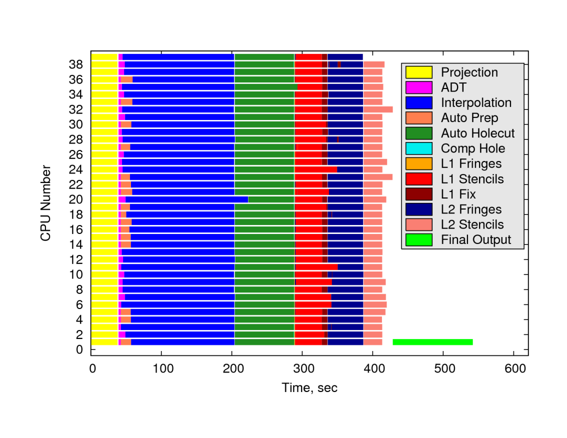
Parallelization

- Code is composed of many tasks
  - Projection, ADT, interpolation, hole-cutting, level-1 interpolation, level-2 interpolation, etc
  - Most tasks are independent of each other
  - Each task reads all input from disk files and writes all results to disk files
- Parallelization uses Message-Passing-Interface (MPI)
  - One master process to distribute and monitor the work
  - Many worker processes, one per CPU
- Reliably reproduces results of the serial code
- The larger the grid system, the better the parallel scaling

Performance of Projection Algorithm

Space Launch System: 892 zones, 375 million points

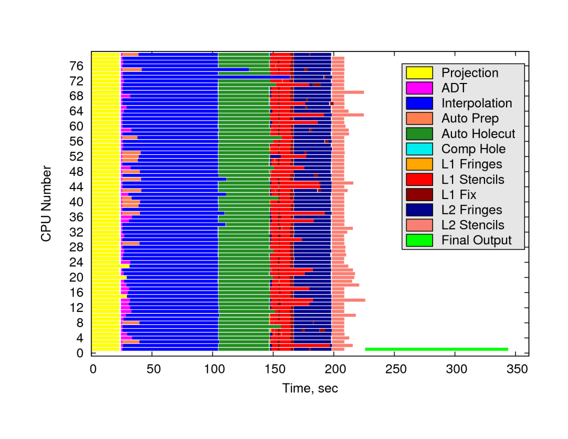
- Wallclock-time to create overset, sec:
- 40 Cores: 544 sec



Performance of Projection Algorithm

Space Launch System: 892 zones, 375 million points

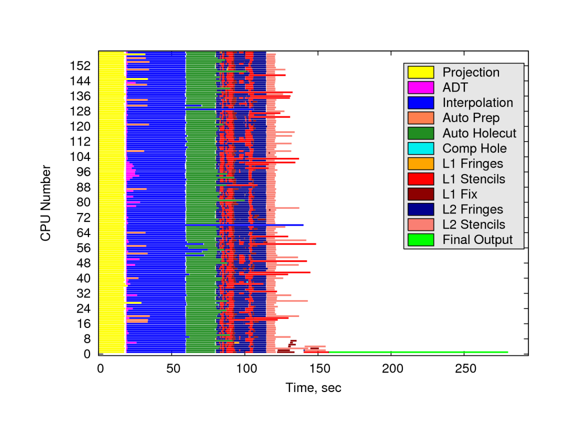
- Wallclock-time to create overset, sec:
- 40 Cores: 544 sec
- 80 Cores: 349 sec



Performance of Projection Algorithm

Space Launch System: 892 zones, 375 million points

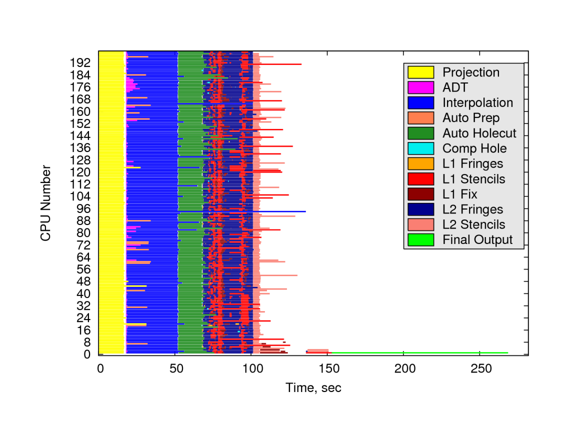
- Wallclock-time to create overset, sec:
- 40 Cores: 544 sec
- 80 Cores: 349 sec
- 160 Cores: 285 sec



Performance of Projection Algorithm

Space Launch System: 892 zones, 375 million points

- Wallclock-time to create overset, sec:
- 40 Cores: 544 sec
- 80 Cores: 349 sec
- 160 Cores: 285 sec
- 200 Cores: 277 sec



Asymptotic performance: 0.74 μsec per grid-pt

Asymptotic perf excluding I/O: 0.43 μsec per grid-pt

Restarting

- Pegasus5 execution consists of many individual tasks
- Each task has a defined set of dependencies (inputs) that are stored in disk files
- Each task results in one or more output files
- Automatically determines which tasks are out of date based on internal time-stamps of input and output files
    - Internal time-stamps written as first and last record in each disk file
    - An incomplete or inconsistent file is considered out of date
- Upon execution Pegasus5 first checks all files and determines which tasks need to be run
- Can successfully restart for:
    - Modifications to user inputs or zones
    - Addition of new zones
    - Incomplete previous run or computer crash
- Allows incremental buildup of complex configurations

Pegasus5 Inputs

Required Inputs

- Standard input file, namelist format
- Volume grids in individual files: X_DIR/name1.x, X_DIR/name2.x, ..., X_DIR/nameN.x

Tools helpful in generating input:

- **peg_setup** script: converts Overflow input file and multi-zone grid file
- Chimera Grid Tools scripts: BuildPeg5i
- Overgrid

Pegasus5 Input File Example

```
$GLOBAL
  FRINGE = 2, OFFSET = 1,
$END

$MESH NAME = "body",
      KINCLUDE= 2, -2, LINCLUDE= 2, -1 OFFSET=2, $END

$MESH NAME = "bodynose",
      JINCLUDE= 2, -1, LINCLUDE= 2, -1, $END

$MESH NAME = "wing", $END

$MESH NAME = "wingcap", $END
```

Pegasus5 Input File Example, continued

```
$BCINP ISPARTOF =     "body",
   IBTYP =     5,   17,   17,   15,
   IBDIR =     3,    2,   -2,   -1,
   JBCS  =     1,    1,    1,   -1,
   JBCE  =    -1,   -1,   -1,   -1,
   KBCS  =     1,    1,   -1,    1,
   KBCE  =    -1,    1,   -1,   -1,
   LBCS  =     1,    1,    1,    1,
   LBCE  =     1,   -1,   -1,   -1,
   YSYM  = 1,
   $END
```

Important Boundary Condition Types

See Pegasus5 manual for complete list

- IBTYP = 1-4: inviscid walls
- IBTYP = 5-8: viscous walls
- IBTYP = -1: Dummy solid wall: used as a hole-cutting wall and designated as an overset boundary
- IBTYP = 10: O-grid periodic face (apply to one face)
- IBTYP = 11-13: Symmetry in X, Y, Z with reflection plane
- IBTYP = 17: Symmetry without reflection plane
- IBTYP = 20: Point-wise block (cell-centered only)
- IBTYP = 21: 2D in Y-direction (apply to one face)
- IBTYP = 22: Axisymmetric in Y (apply to one face)
- IBTYP = 51-53: C-grid flow-through (apply to one face)
- IBTYP = not listed above: other boundary condition

- All zonal boundaries not assigned an IBTYP are treated as overset outer boundaries

Pegasus5 Execution

- Once you have the input file and the volume grids are installed in the X_DIR directory you can execute the code:
- Serial version:

```
pegasus5 < peg.i
```

- MPI Parallel version using N cpus:

```
mpiexec -np N pegasus5mpi < peg.i
      <or>
mpiexec -np N pegasus5mpi -ifile peg.i
```

- Note: mpi version requires that all CPUs have access to the same copy of the working directory

Pegasus5 Output

- Pegasus5 creates **WORK** directory, contains all of the intermediate output files created by Pegasus5
  - Typically no need to examine or read these files directly
  - In order to re-run a case from the beginning, simply remove **WORK**
- All informational output written to a file named **log.mmdd.hhmm**, examine this file to see what Pegasus5 did
- Note: the mpi version will create individual log files for each process, named log.mmdd.hhmm.XXXX, these will be concatenated together upon successful completion of the run
- The XINTOUT file contains all of the interpolation stencils and blanking information used by the flow solver

Post Execution Steps

- Examine log file and verify successful completion
- Use **peg_plot** with option 3 to create composite grid file
- Examine minimum hole cuts and make sure no active points are left inside a solid body
  - Plot hole boundaries in plot3d, function 2
  - Plot grid slices in overgrid, tecplot, fieldview, etc
  - Plot orphan points in plot3d, overgrid, tecplot, etc
  - Look for orphan points left inside a solid body
- Examine and eliminate cause of orphan points

End of log file: Stencils and Orphans

```
-------------------------------------------------------------------------------
                    |               |               |                         |
Mesh Name           |Interpolated   |Interpolation  |Orphan Points            |
                    |Boundary Points|Stencil        |                         |
                    |               |               |                         |
-------------------------------------------------------------------------------
                    |               |               |                         |
fuselage            |Level 1:  10634|Level 1:  32934|1st Fringe:      0       |
                    |Level 2:  24578|Level 2:  10498|2nd Fringe:      0(Fixed)|
                    |Total:    35212|Total:    43432|Total:           0       |
                    |               |               |                         |
-------------------------------------------------------------------------------
                    |               |               |                         |
wing                |Level 1:  38609|Level 1:  30486|1st Fringe:      2       |
                    |Level 2:  49279|Level 2:  12261|2nd Fringe:      0(Fixed)|
                    |Total:    87888|Total:    42747|Total:           2       |
                    |               |               |                         |
-------------------------------------------------------------------------------
                    |               |               |                         |
wingcap             |Level 1:  20251|Level 1:  12491|1st Fringe:      0       |
                    |Level 2:    242|Level 2:  22748|2nd Fringe:      0(Fixed)|
                    |Total:    20493|Total:    35239|Total:           0       |
                    |               |               |                         |
-------------------------------------------------------------------------------
                    |               |               |                         |
Grand Total         |Level 1: 262641|Level 1: 262641|1st Fringe:     14       |
                    |Level 2: 267467|Level 2: 267467|2nd Fringe:      0       |
                    |Total:   530108|Total:   530108|Total:          14       |
                    |               |               |                         |
-------------------------------------------------------------------------------
```

End of log file: Execution Time

PROCESS	CPU(sec)	WALL(sec)	Sub-procs	Max sub-proc(sec)
projection	13.875	2.328	122	1.672
adt	4.656	1.266	13	0.906
interpolate	65.922	24.586	122	3.867
auto_hbound	66.438	26.898	3	26.906
man_hbound	0.000	0.000	0	0.000
auto_cut	42.234	4.906	30	4.805
man_cut	0.000	0.000	0	0.000
comp_hole	1.156	0.141	13	0.125
spec_int1	0.508	0.055	13	0.062
spec_level1	8.078	0.859	13	0.867
level1fix	1.734	2.305	1	1.734
spec_int2	19.859	2.195	61	0.930
spec_level2	9.859	1.023	13	1.016
xintout	1.469	1.477	1	1.469

SUM of PROCESS TIME for all processes (secs): 235.789

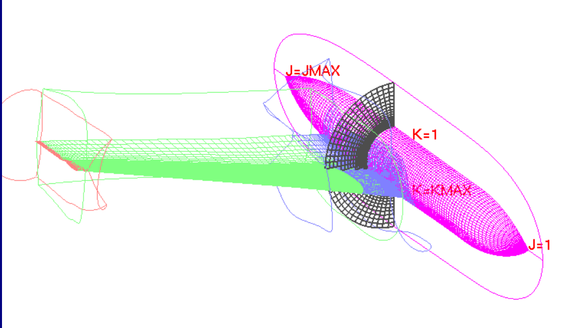
ELASPSED WALL TIME(secs): 37.703

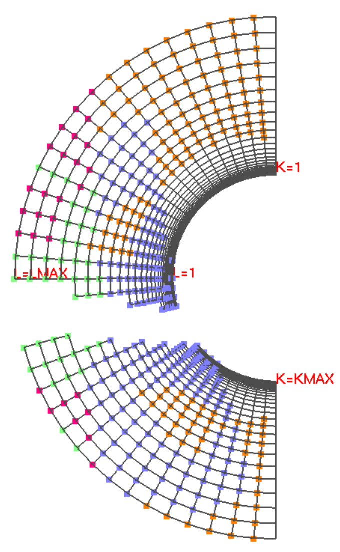
EXECUTION SPEED-UP = 6.25 using 15 processors.

Output: peg_plot

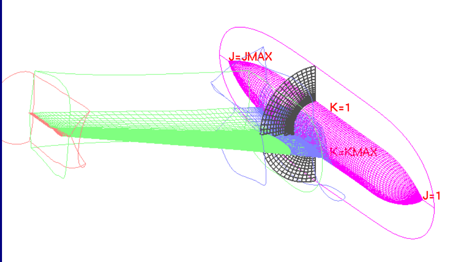
- Grid file: use the peg_plot program to create the grid file used by the flow solver, and to plot and check the results of the Pegasus5 run
  - Use peg_plot option 3 first to examine the results of the hole cutting
  - The peg_plot option 2 (or option 1) to blank out the higher-level fringes in the resulting grid file
  - This illustrates the borders of where information is passed between overlapping zones
  - Useful when plotting the flow solution as it minimizes the overlap
- Note: Overflow does not use the iblank array in the grid file, so any peg_plot option works when creating the grid file that will be passed to Overflow

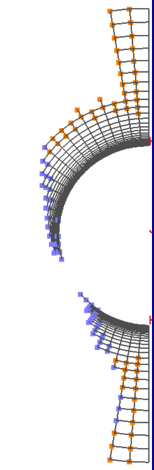
Example: peg_plot Option 3





- Wing-body example using peg_plot option 3
- View the fuselage zone in overgrid
- Shows auto hole cut by the wing
- Fringe points shown with colored symbols

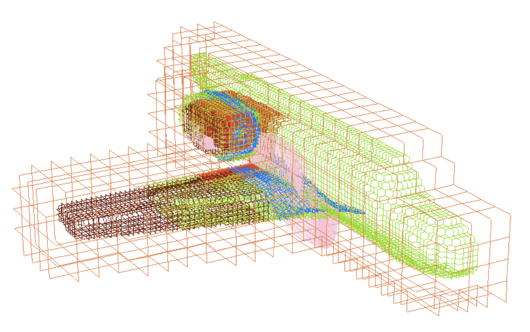




- Wing-body example using peg_plot option 2
- Higher-level fringe points have been blanked out
- Shows the virtual overlap after the Level-2 interpolation
- Flow-solver still keeps the higher-level fringes active: they can be used as donor cells for other zones

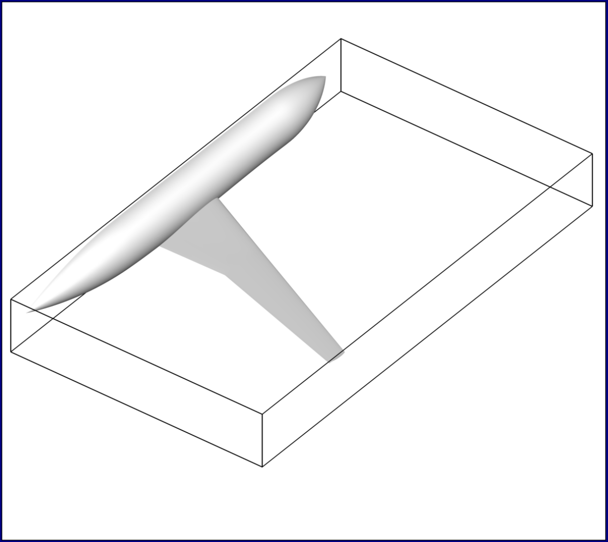
Examining the Hole Cuts

- Use plot3d function 2: plots the outlines of the holes
- Use Overgrid, etc: plot slices through grids
- Search log file for "composite hole": lists number blanked points in each mesh
- Use peg_hole_surf to extract grid surfaces used by each $HCUT hole cutter



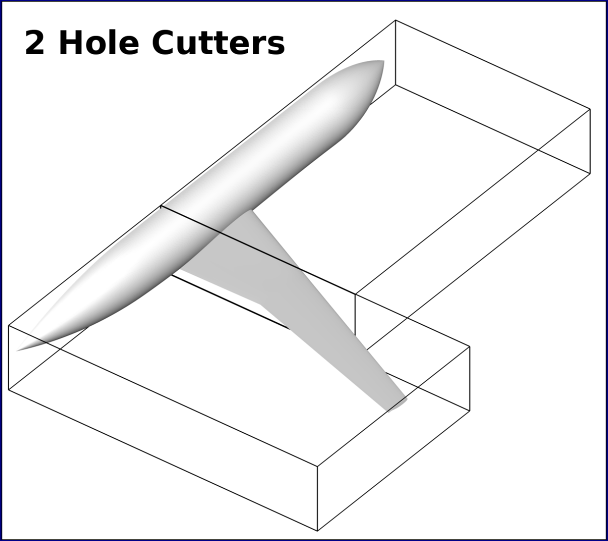
Version 5.2: Auto Domain Decomposition

Automatic Creation of HCUT Hole-Cutters To Fit The Geometry



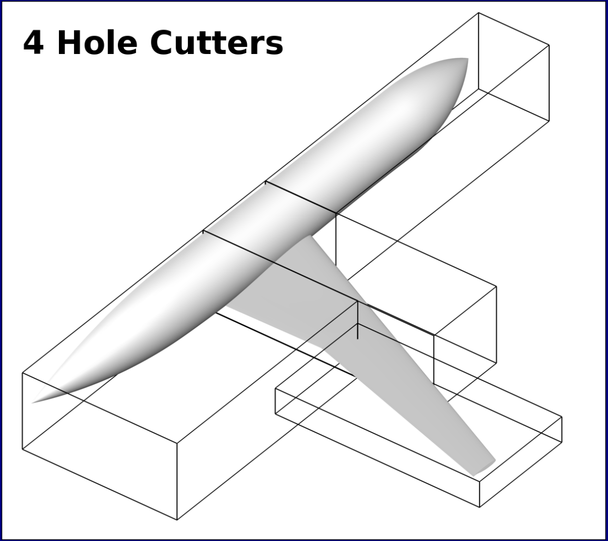
Version 5.2: Auto Domain Decomposition

Automatic Creation of HCUT Hole-Cutters To Fit The Geometry



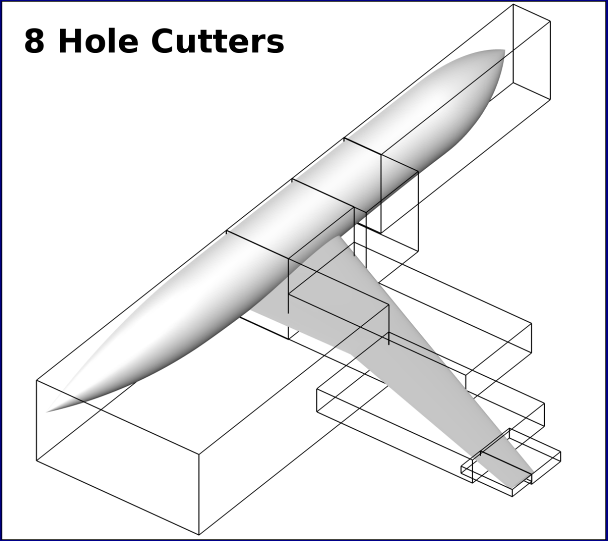
Version 5.2: Auto Domain Decomposition

Automatic Creation of HCUT Hole-Cutters To Fit The Geometry



Version 5.2: Auto Domain Decomposition

Automatic Creation of HCUT Hole-Cutters To Fit The Geometry



Version 5.2: Auto Domain Decomposition

CUTTER_CONTROL Namelist

- AUTOHCT > 1: Automatically domain splitting
- Optional:
  - Use XCUTS,YCUTS,ZCUTS to exclude splitting in X, Y, Z directions
  - Use CUTPLDIR,CUTPLVAL to further control splitting procedure

```
$CUTTER_CONTROL
    AUTOHCT = 8,
    CNX = 512,
    CNY = 512,
    CNZ = 512,
    XCUTS = .TRUE.,
    YCUTS = .TRUE.,
    ZCUTS = .TRUE.,
    CUTPLDIR = $<list-of-cut-pl
    CUTPLVAL = $<list-of-cut-pl
    $END
```

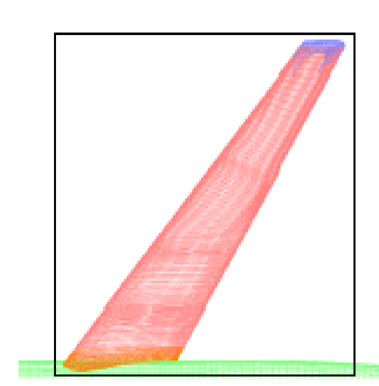
Custom Hole Cutting

- The $HCUT namelists are used to define separate hole-cutters
- Without any $HCUT namelists in input file, pegasus5 uses ALL solid-wall surfaces to cut holes from ALL zones
- Adding an $HCUT entry eliminates this default hole-cutter
- Adding multiple $HCUT entries increases resolution and parallel efficiency

```
$HCUT NAME = "hcutter1",
    MEMBER  = "body1",
              "body2",
    INCLUDE = "bodynose",
              "wing",
              "wingcol",
    CNX = 512,
    CNY = 512,
    CNZ = 512,
    CARTX = -100.0, 100.0,
    CARTY = -50.0, 50.0,
    CARTZ = 0.0, 100.0,
    $END
```

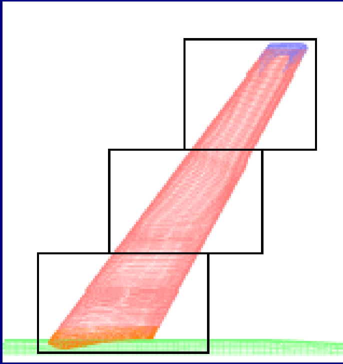
Custom Hole Cutting

```
$HCUT NAME = "winghole",
    MEMBER  = "wing",
              "wingcol",
              "wingcap",
              "body",
    INCLUDE = "body",
              "wingbox",
              "bodybox",
              "farbox",
    CARTX = 100.0, 400.0,
    CARTY = 10.0, 150.0,
    $END
```

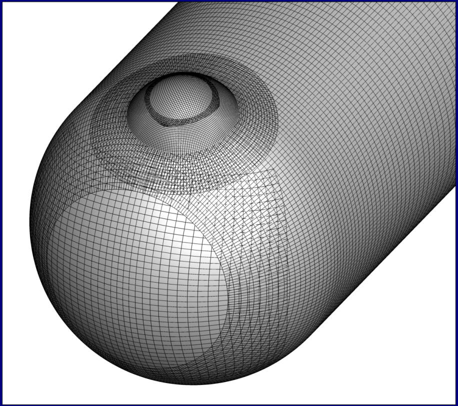


Custom Hole Cutting

```
$HCUT NAME = "winghole1",
    MEMBER  = "wing", "wingcol", "body",
    INCLUDE = "body", "wingbox",
              "bodybox", "farbox",
    CARTX = 100.0, 250.0,
    CARTY = 10.0, 51.0,
    $END
$HCUT NAME = "winghole2",
    MEMBER  = "wing",
    INCLUDE = "wingbox", "farbox",
    CARTX = 200.0, 350.0,
    CARTY = 50.0, 101.0,
    $END
$HCUT NAME = "winghole3",
    MEMBER  = "wing", "wingcap",
    INCLUDE = "wingbox", "farbox",
    CARTX = 240.0, 400.0,
    CARTY = 100.0, 150.0,
    $END
```

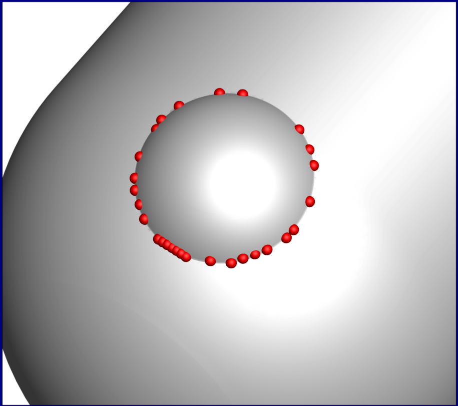


```
$HCUT NAME = "NewHole1",
    MEMBER  = "body", "bump",
    INCLUDE = "body",
    CARTX = 0.0, 1.0,
    CARTY = 0.0, 1.0,
    CARTZ = 0.0, 1.0,
    OCORNER = 00001111,
    HOFFSET = 1,
    $END
```



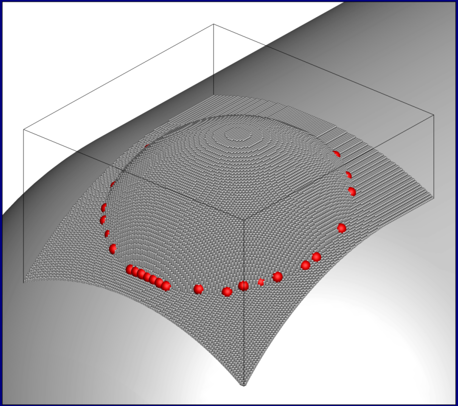
Version 5.2b: Protuberance HCUT

```
$HCUT NAME = "NewHole1",
    MEMBER  = "body", "bump",
    INCLUDE = "body",
    CARTX = 0.0, 1.0,
    CARTY = 0.0, 1.0,
    CARTZ = 0.0, 1.0,
    OCORNER = 00001111,
    HOFFSET = 1,
    $END
```



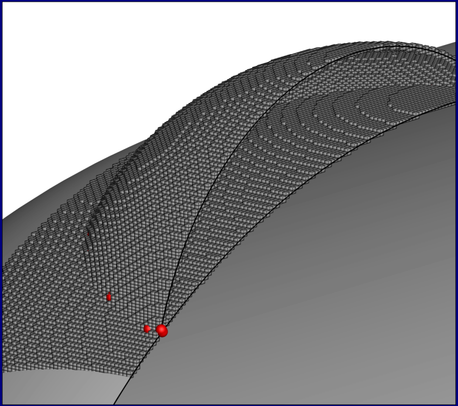
Version 5.2b: Protuberance HCUT

```
$HCUT NAME = "NewHole1",
    MEMBER  = "body", "bump",
    INCLUDE = "body",
    CARTX = 0.0, 1.0,
    CARTY = 0.0, 1.0,
    CARTZ = 0.0, 1.0,
    OCORNER = 00001111,
    HOFFSET = 1,
    $END
```



Version 5.2b: Protuberance HCUT

```
$HCUT NAME = "NewHole1",
    MEMBER  = "body", "bump",
    INCLUDE = "body",
    CARTX = 0.0, 1.0,
    CARTY = 0.0, 1.0,
    CARTZ = 0.0, 1.0,
    OCORNER = 00001111,
    HOFFSET = 1,
    $END
```



Hole-Cutting Troubleshooting

No holes cut due to leaks or gaps in solid-wall surfaces

- Use CARTX,CARTY,CARTZ to seal gap
- Use PHANTOM zone to seal gap
- Edit input file and extend solid-wall boundary

Holes too small near thin bodies, such as TE of a thin wing

- Increase OFFSET or HOFFSET to enlarge holes
- Increase CNX, CNY, CNZ to improve resolution

Hole points not cut properly near collar grids

- Increase OFFSET or HOFFSET to enlarge holes

Holes cut in solid walls in regions of high curvature

- Increase grid resolution
- Use $REGION and $VOLUME namelists to unblank holes

Manual Hole Cutting

Manual hole-cutting from pegsus4 retained in Pegasus5

$BOUNDARY/$SURFACE namelists

Can specify a group of zonal surfaces which will cut holes in the specified zones

$BOUNDARY/$BOX namelists

Can specify a range of x,y,z coordinates of a box which will cut holes in the specified zones

$REGION/$VOLUME namelists

Can specify a range of j,k,l zonal indices to create a hole, or to unblank part of an existing hole

Orphan Points

- Orphan points are fringe points for which no valid interpolation donor can be found
- 2nd- and 3rd-layer fringe-point orphans are reset to active interior points by default
- Overflow will update the solution data at orphan points by averaging the neighboring grid points
    - A few isolated orphan points are usually acceptable, but it is advisable to find and fix most or all orphans
    - Orphans on solid-wall surfaces usually indicate a serious problem with surface resolution or projection, and should be fixed

Identifying Orphan Points

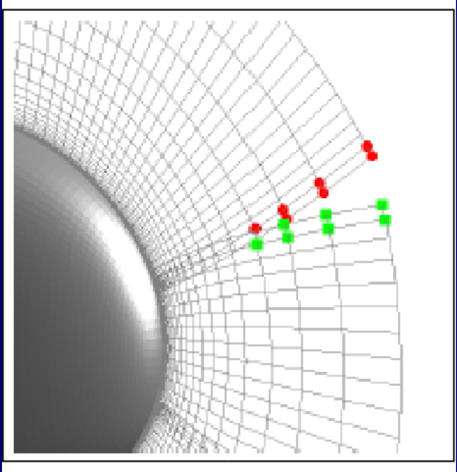
- Orphans are reported in the log file, in the output of **peg_plot** and using the **peg_orph** program
- Use the **peg_plot** program to create a grid file with iblank
- Can plot orphans using several programs:
  - **plot3d:** Plot using function 3
  - **overgrid:** Grid Diagnostics module
  - **tecplot:** Use contours or scatter plots

Causes of Orphans

- Bad hole cuts
- Insufficient overlap
- Poorly resolved geometry in regions of surface curvature
- Inappropriate or missing boundary conditions

Insufficient Overlap

- Increase surface-grid overlap
- Add more splaying to volume-grid generation
- Add a Cartesian box to cover the open space
- Grow outer boundaries further
- Add more grid points



Utility Codes

- **peg_setup:** creates input file and grid files
- **peg_hole_surf:** created plot3d grid files containing solid-wall surfaces used in automatic hole cutting
- **peg_plot:** creates composite plot3d grid file with iblank
- **peg_plot_center:** creates composite plot3d grid file with iblank for cell-centered grids
- **peg_diag:** produces diagnostic file for plotting interpolation parameters and connection information
- **peg_orph:** lists orphan points for each zone
- **peg_proj:** creates diagnostic plot3d files to plot maximum surface projections
- **XINtegrity:** Runs tests and verifies integrity of the XINTOUT file

New Utility Codes Available in Version 5.2

- **dcintegrity:** Runs tests and verifies integrity of the dci file
- **hcut_info:** Prints data about each HCUT hole cutter
- **hcut_plot:** Creates plotting files used to examine HCUT hole cutters
- **peg5_plotcpu.pl** Creates plot of CPU usage for each processor for an pegasus5mpi run

Summary

- Pegasus5 successfully automates most of the oversetting process
  - Dramatic reduction in the user input over previous generations of overset software
  - Reduced complex-geometry oversetting time from weeks to hours
  - Significant reduction in user-expertise requirements
- Not a “black-box” procedure: care must be taken to examine the resulting grid system
- NASA Software catalog page for Pegasus5:

`https://software.nasa.gov/software/ARC-15117-1A`