



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

Stuart Rogers

NASA Ames Research Center

16th Overset Grid Symposium Tutorial

Dayton, Ohio
October 7, 2024



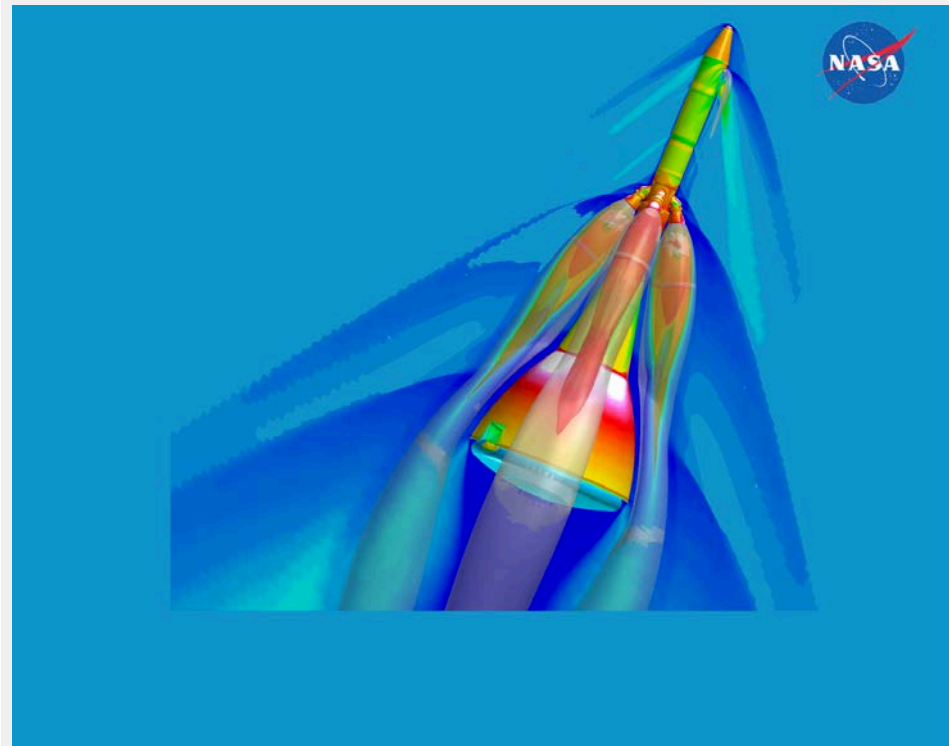
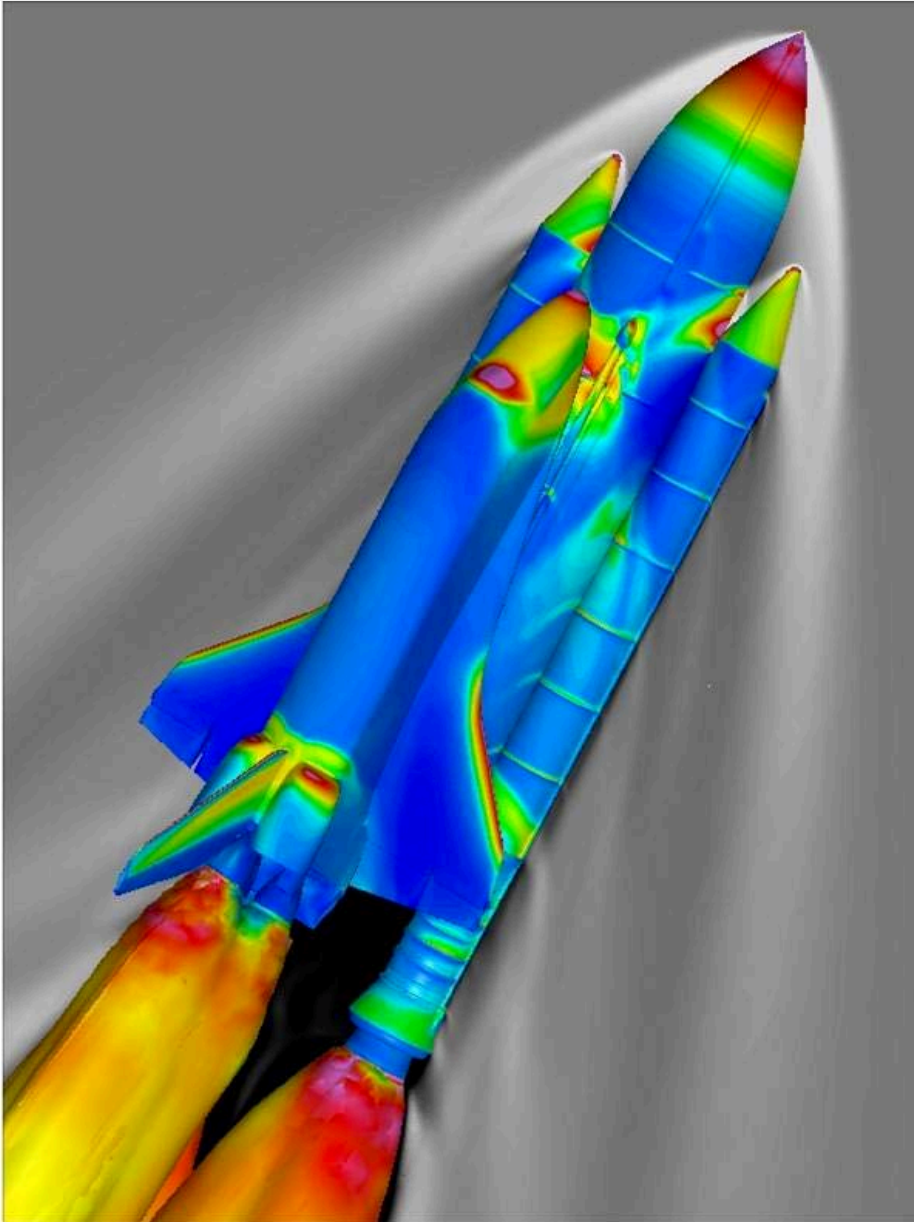
Acknowledgements

- 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
 - NASA Space Launch System

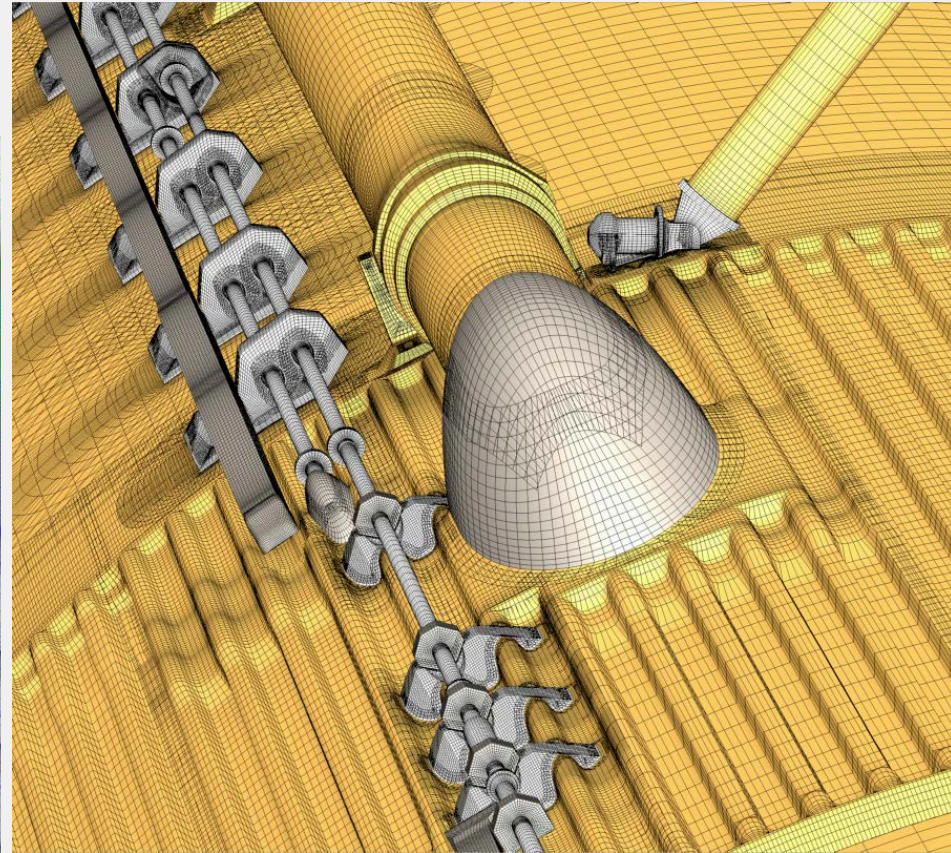
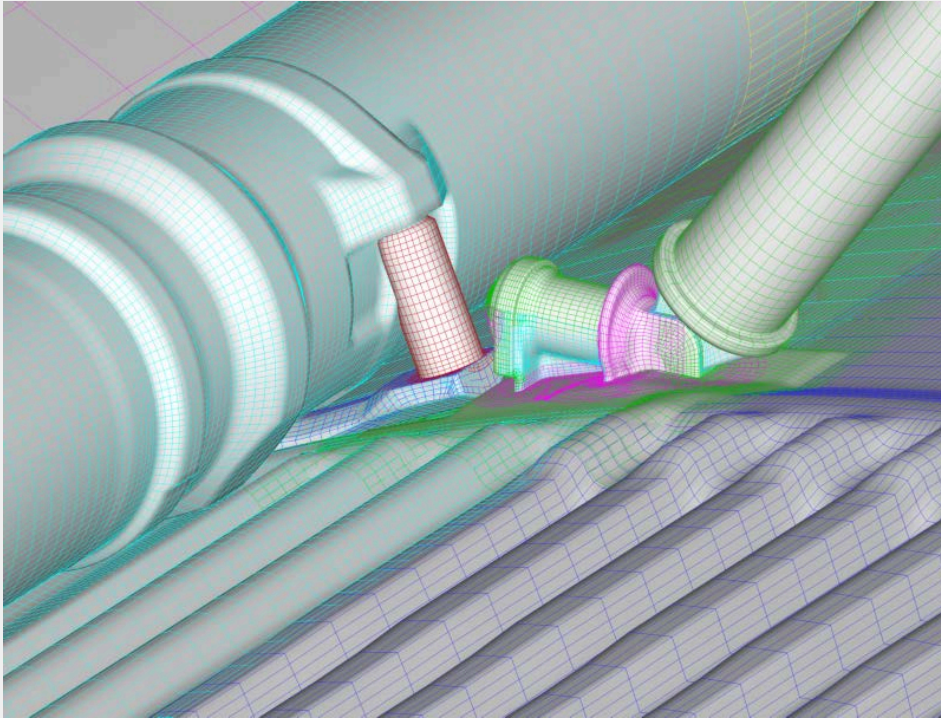
Pegasus5 was co-winner of the 2016 NASA Software of the Year Award

1. **Introduction and Background**
 - The challenge
 - Structured vs unstructured CFD
 - Overflow and overset gridding
2. Overset-grid workflow
3. Pegasus5 features and automation
4. Overview of usage
5. Conclusion

The Oversetting Challenge



The Oversetting Challenge

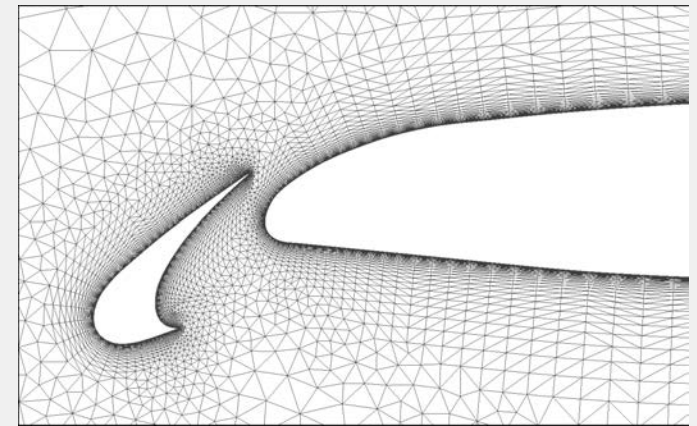
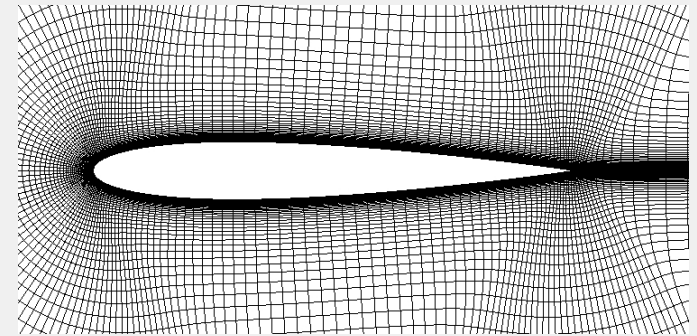


Background

- 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?
 - Simulate actual flight conditions
 - Complex Geometry Capability
 - Run many simulations covering entire flight envelope
 - In-depth investigation of single simulation
 - Very powerful combined with wind-tunnels and flight tests

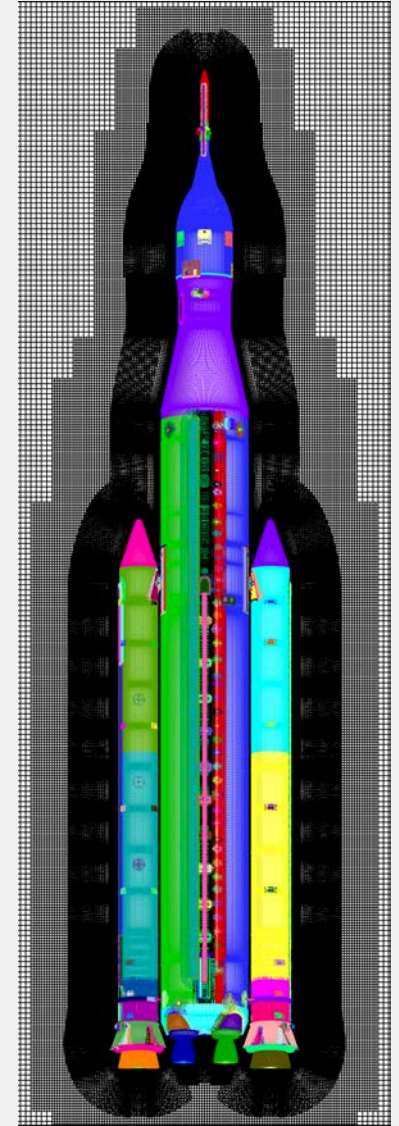
Background: Structured and Unstructured Grids

- 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
 - Requires more computer time and memory

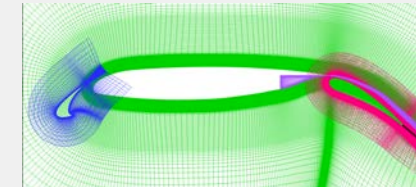
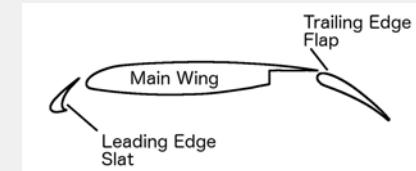
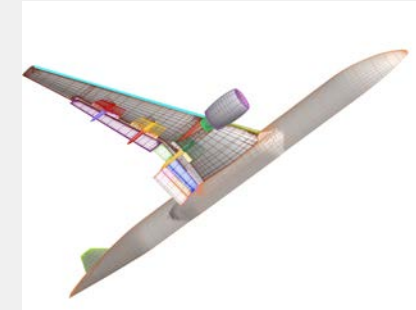


Background: Overflow CFD Flow Solver

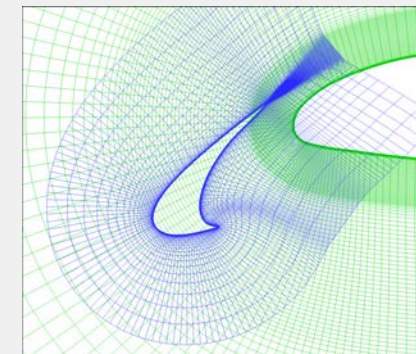
- What is Overflow?
 - NASA developed structured CFD solver
 - One of the most extensively used CFD solvers in NASA
 - Used by every major NASA vehicle program
 - Most users of Pegasus5 use the Overflow solver
- NASA Software Catalog
 - <https://software.nasa.gov/software/LAR-20095-1>



Background: Overset Structured Grids



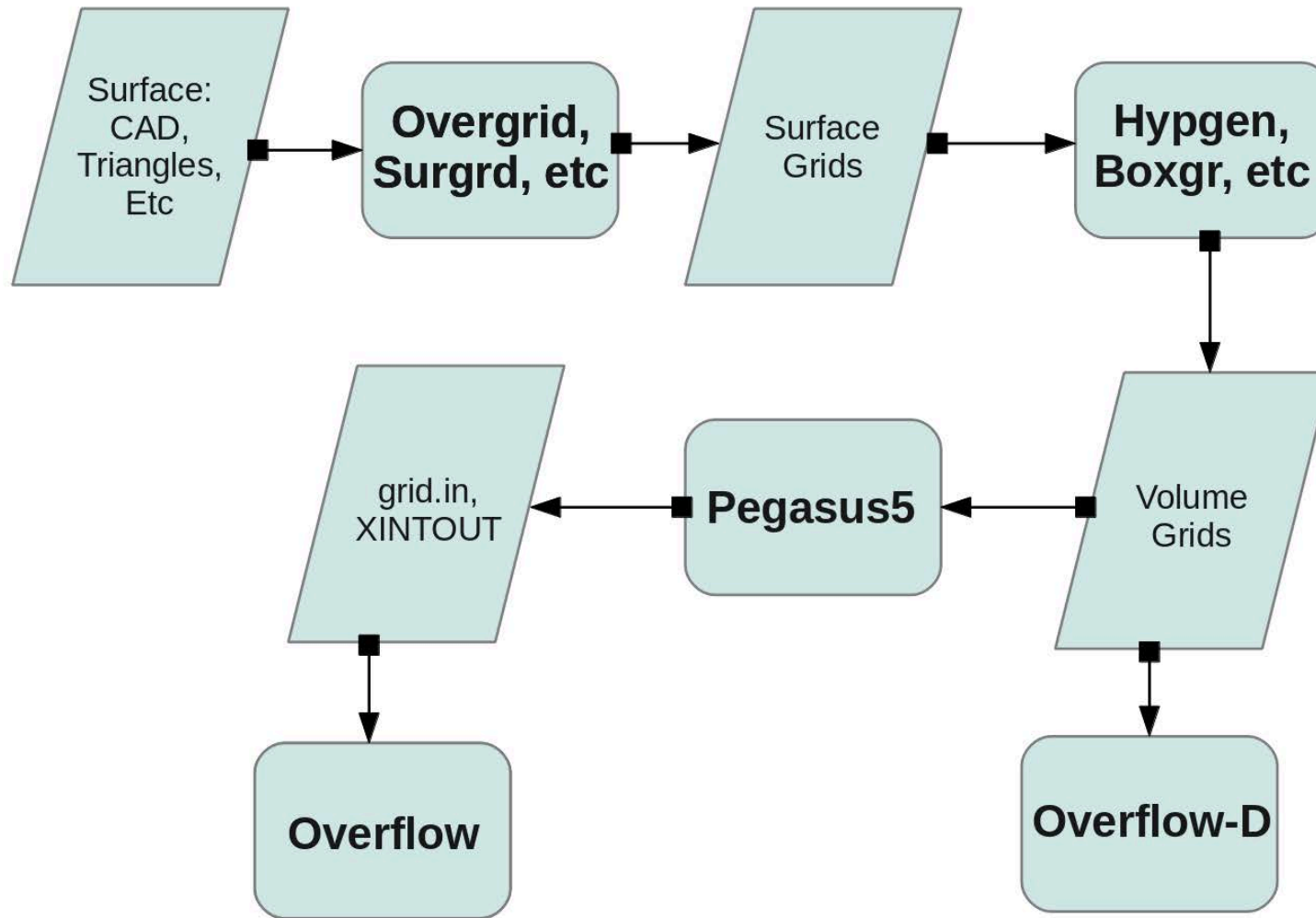
- What are Overset Structured Grids?
 - Split complex geometry problems into simple components
 - Enables highest quality grids in viscous boundary layer
 - Requires software capable of integrating the overlapping grids
 - An entire aircraft requires 100s of millions of grid cells





1. *Introduction and Background*
2. **Overset-grid workflow**
 - Workflow
 - Nomenclature
3. Pegasus5 features and automation
4. Overview of usage
5. Conclusion

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 single 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
- **Field point:** same as **Interior point**
- **Iblank value:** each grid point is assigned an integer value to denote the type of point:
 - $iblack < 0$: fringe point
 - $iblack = 0$: hole point
 - $iblack = 1$: active interior point
 - $iblack = 101$: orphan point

1. *Introduction and Background*
2. *Overset-grid workflow*
3. **Pegasus5 features and automation**
 - Hole cutting
 - Level-2 interpolation to optimize overlap
 - Projection
 - Parallelization
4. Overview of usage
5. Conclusion

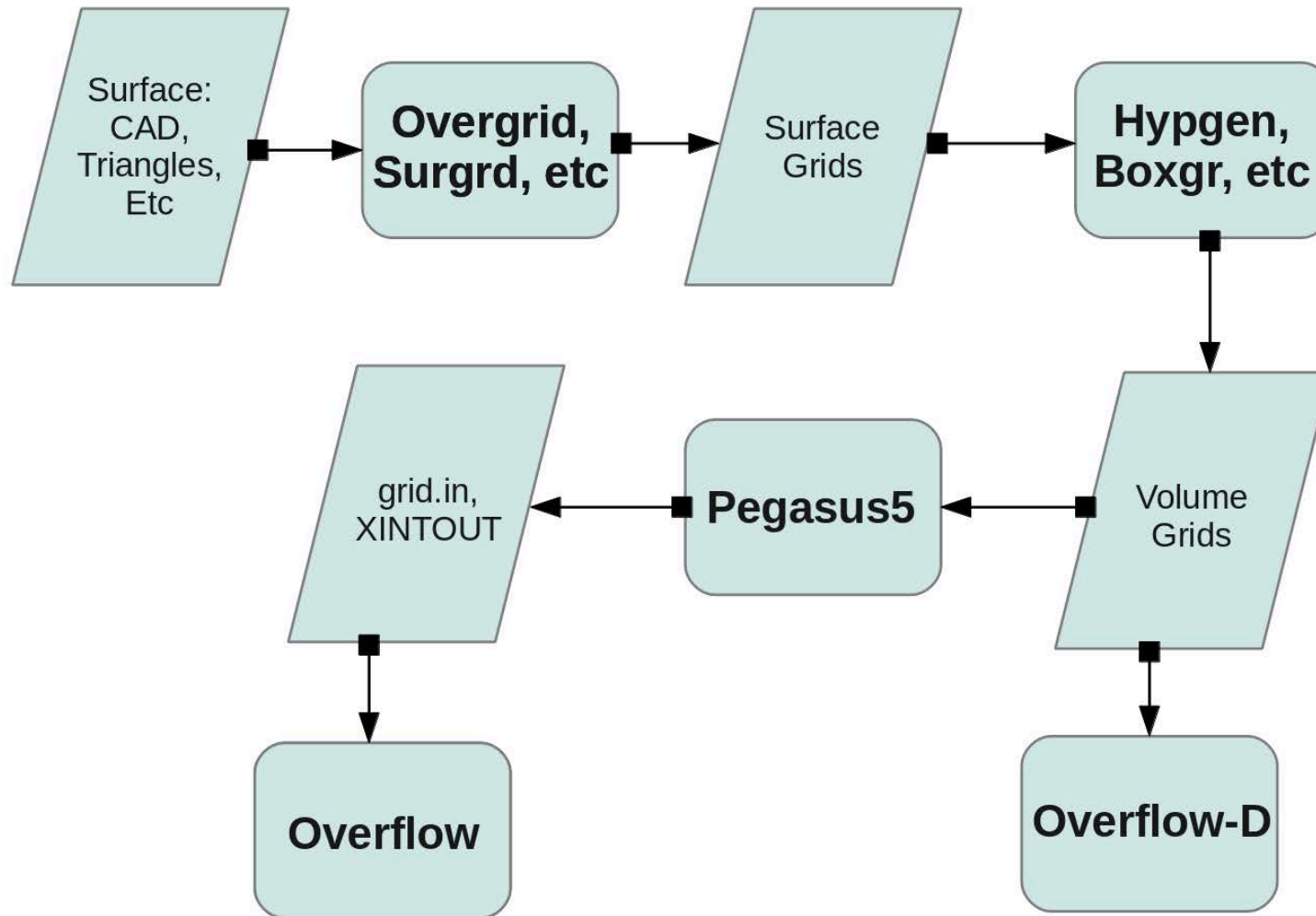
Pegasus5 Goals and Features



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

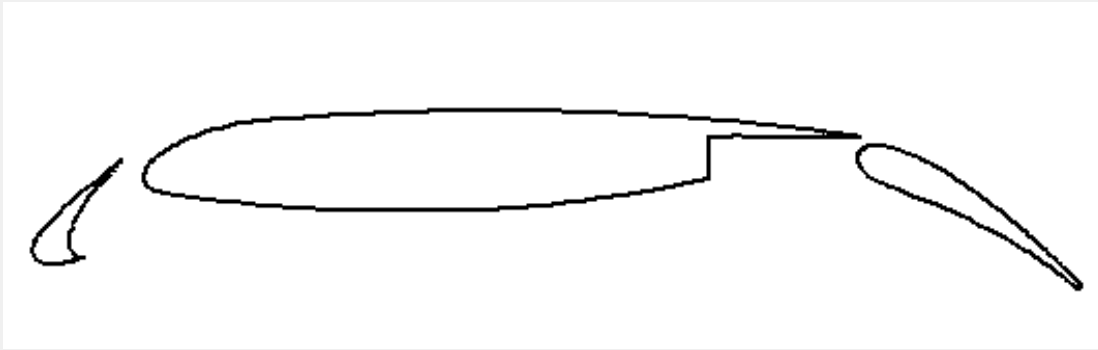
Overset-Grid Workflow For Overflow



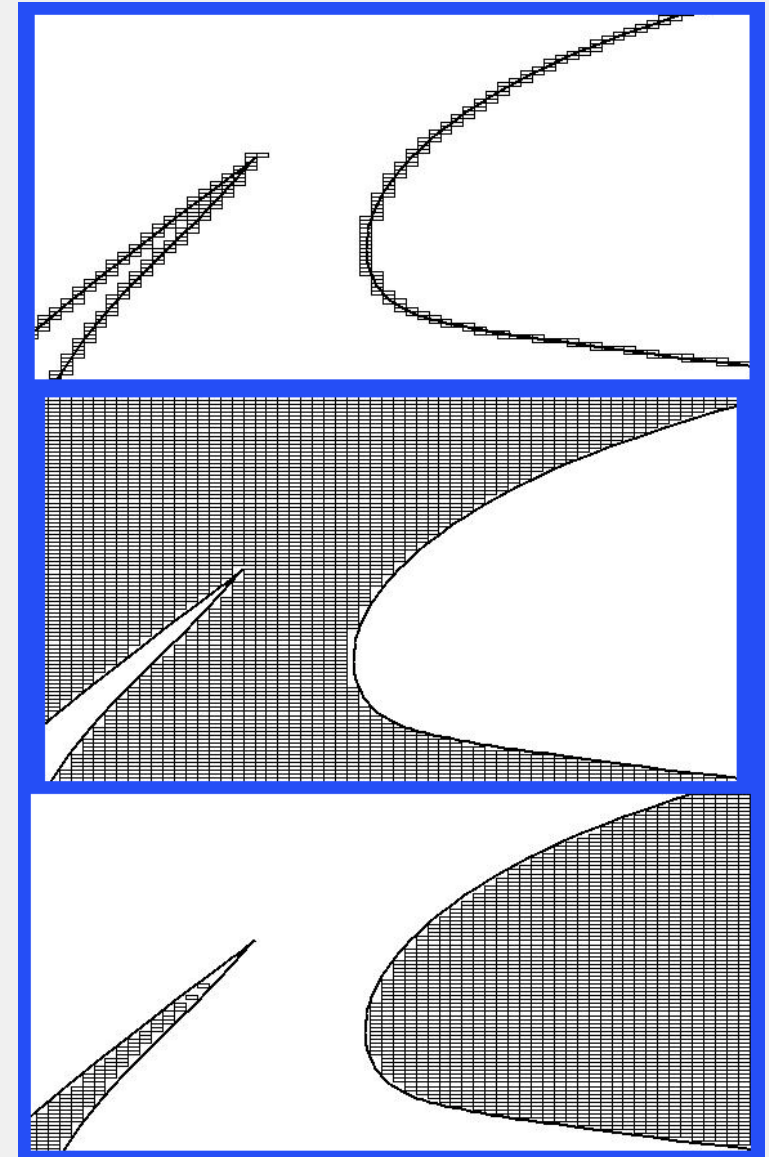


Auto-Hole Cutting Using a Cartesian Map

2D Multi-Element Airfoil Example



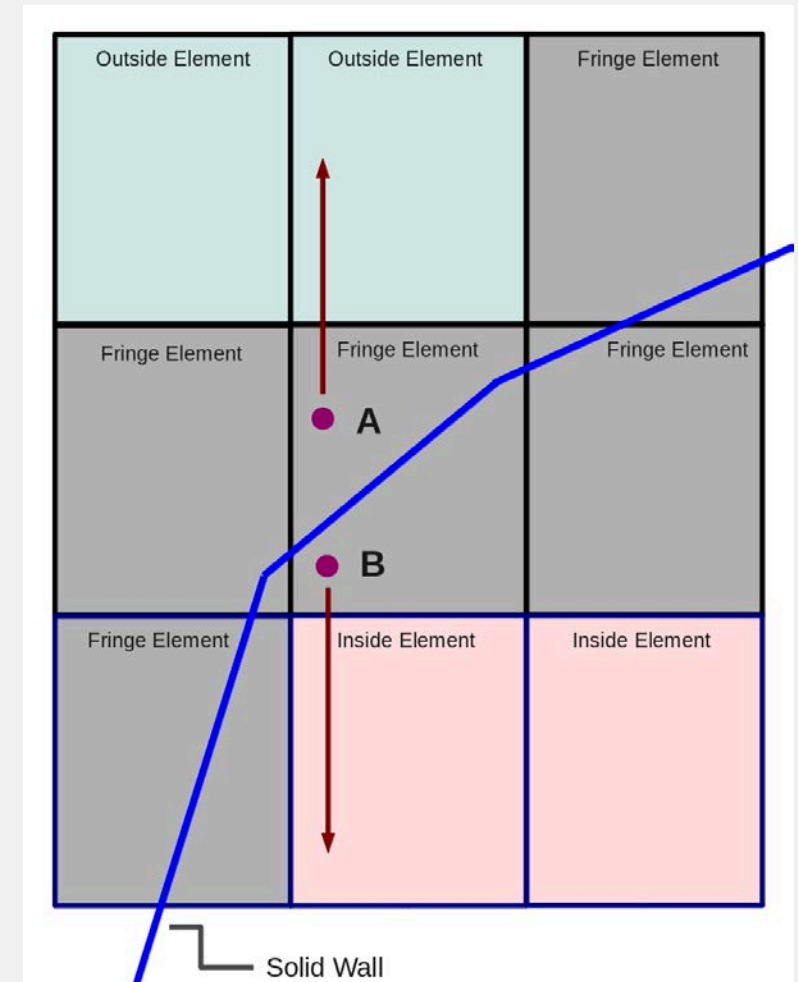
- Identify solid-wall surfaces and overlay with Cartesian map
 - **Fringe elements:** Cartesian elements which intersect walls
 - **Outside elements:** Identify with painting algorithm
 - **Inside elements:** All other elements



Auto-Hole Cutting

Cutting of Candidate Points

- Loop through all volume grid points
- Identify Cartesian cell containing grid point
- 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

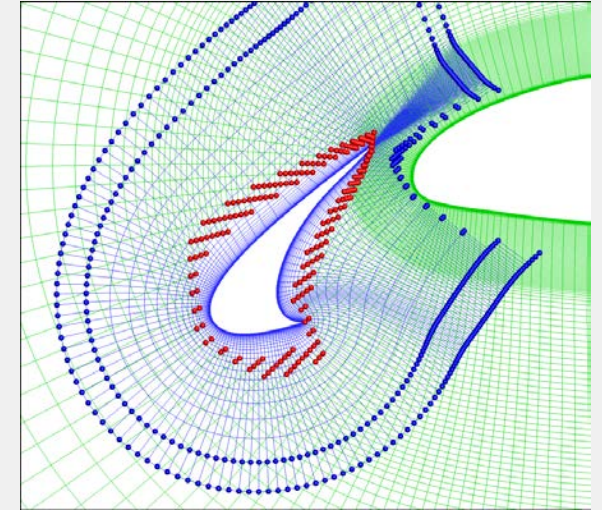


Interpolation Stencil Search

- 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

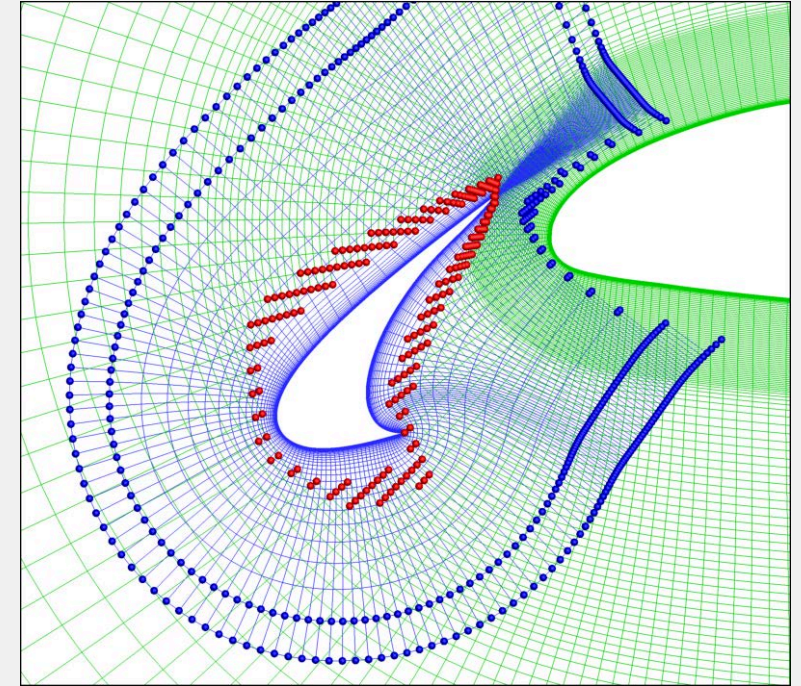
- A fringe point is one which requires updating in the flow-solver via interpolation from a neighboring zone
- Outer-boundary fringe points:
 - All points on the boundary of a zone that do not receive a flow-solver boundary condition are identified as outer-boundary fringe points
- Hole-boundary fringe 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



Two layers of Level 1 Fringes

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



Two layers of Level 1 Fringes

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

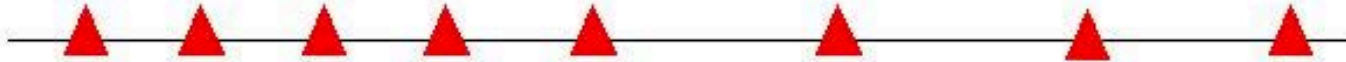
Level-2 Interpolation



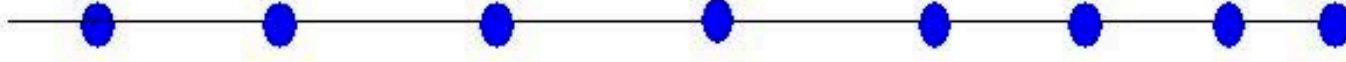
One-Dimensional Example



Mesh A



Mesh B



Mesh C



Level-2 Interpolation

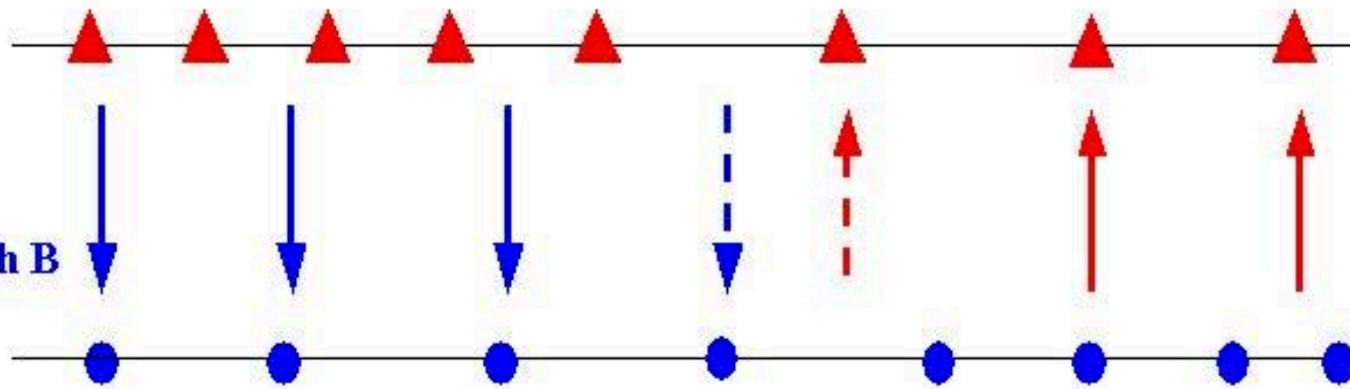


Step 1: Interpolate Between Meshes



Mesh A

Mesh B



Arrow denotes direction of information flow

Level-2 Interpolation

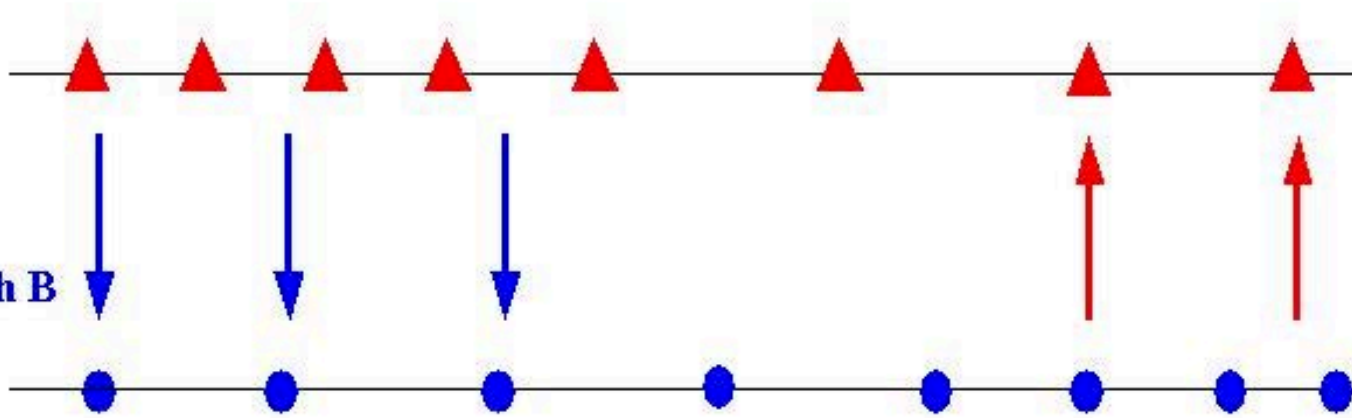


Step 2: Remove Invalid Interpolations



Mesh A

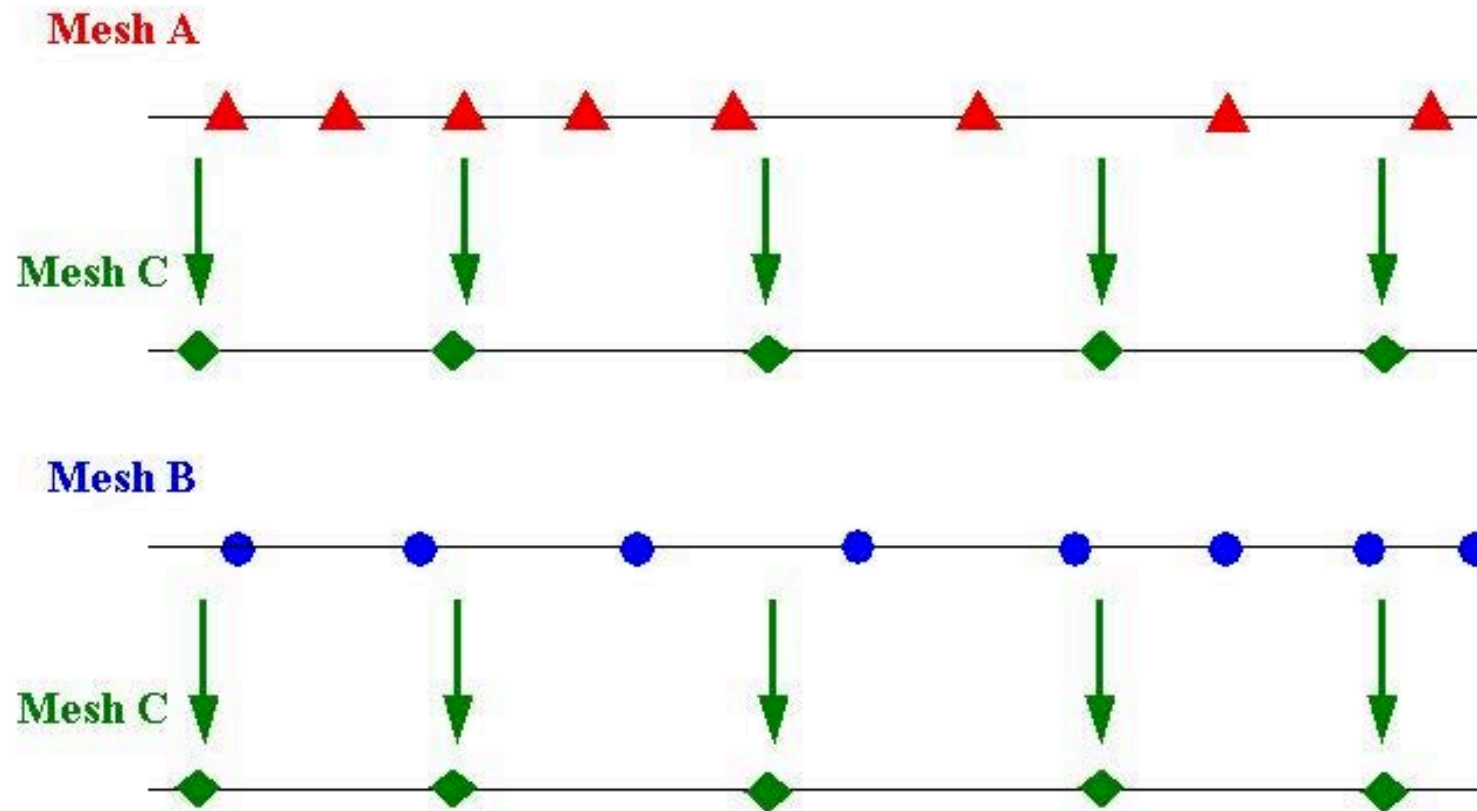
Mesh B



Level-2 Interpolation



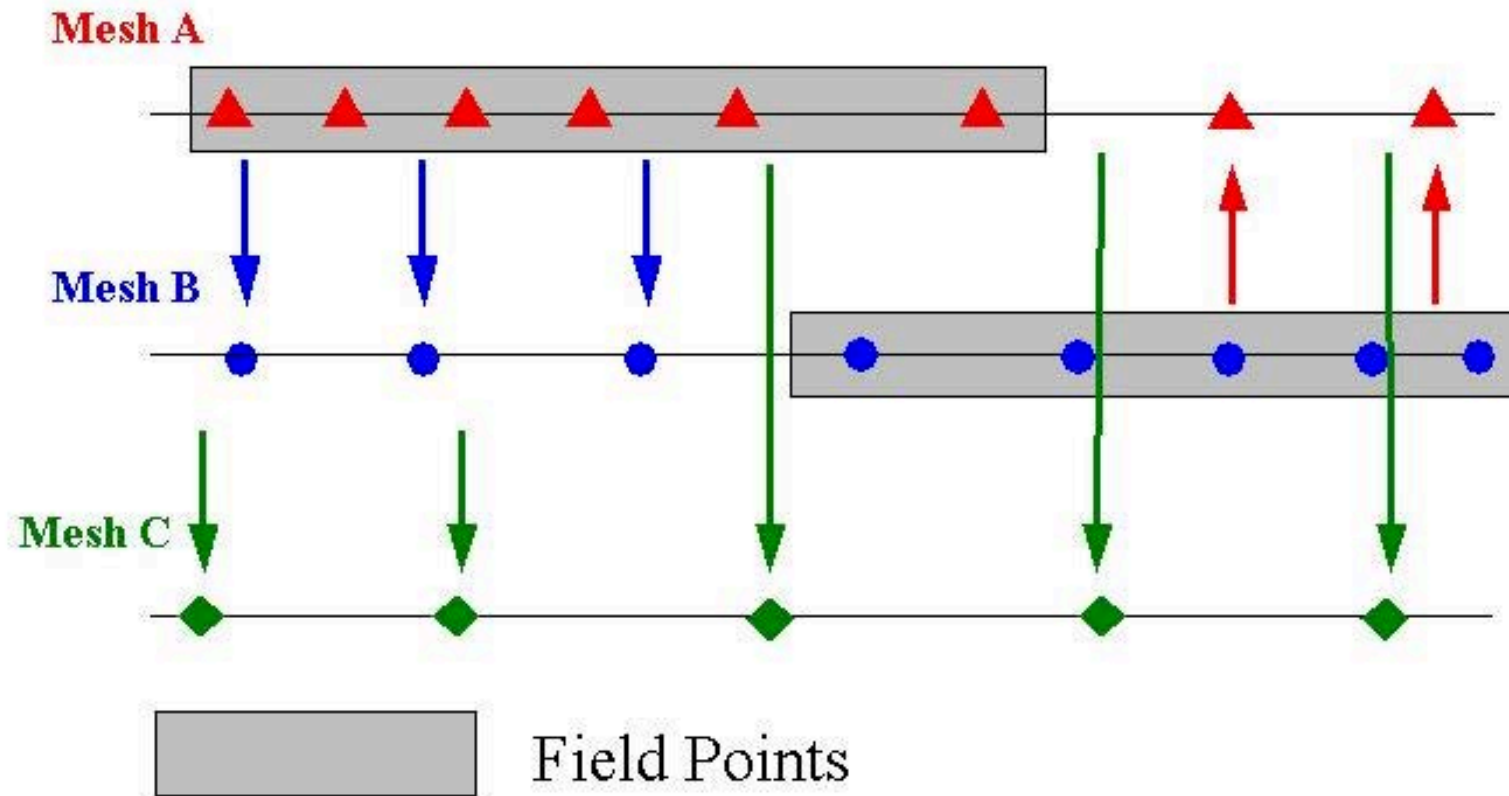
Step 3: Repeat For Other Meshes



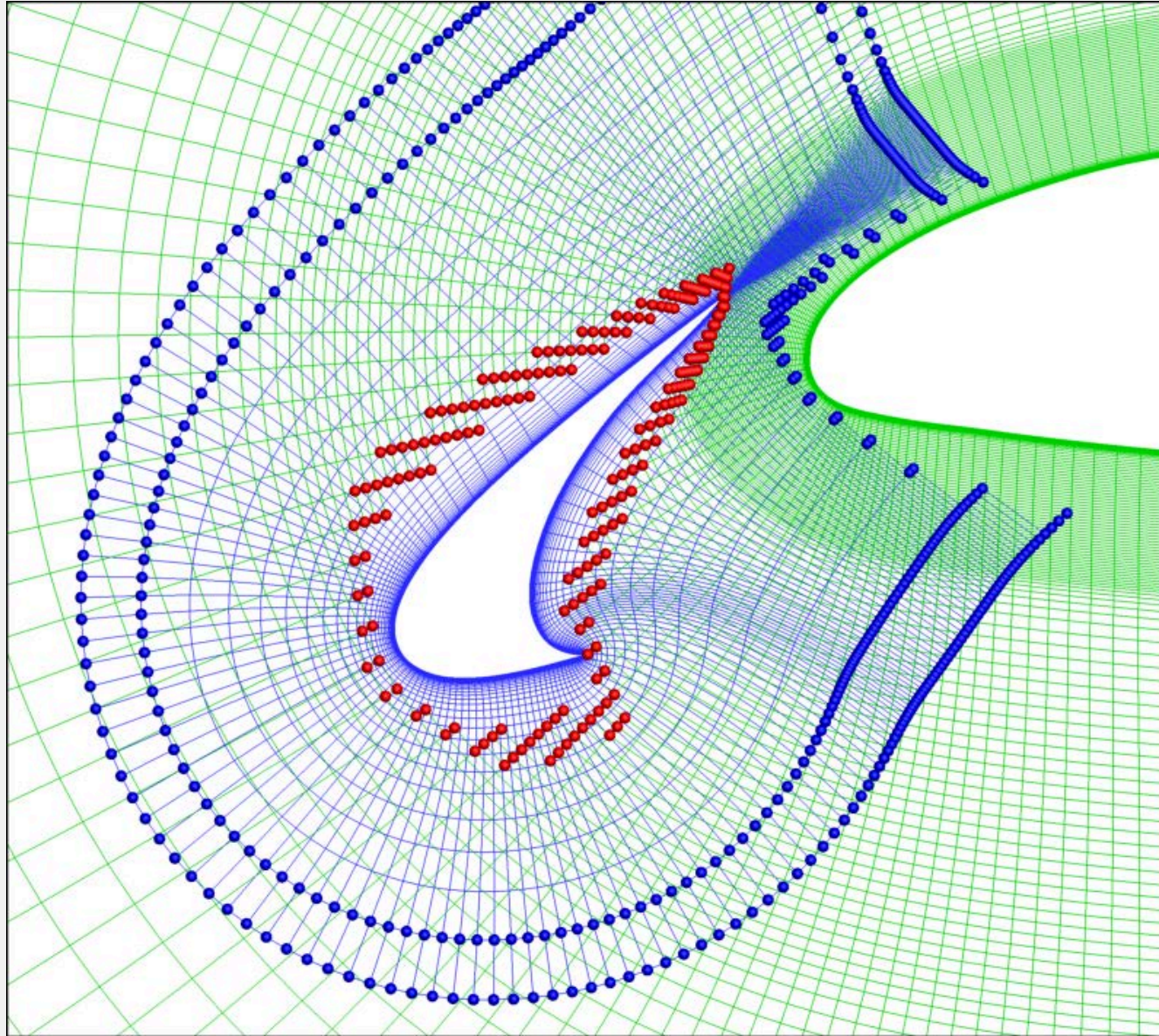
Level-2 Interpolation



Step 4: Keep Finest Mesh Points as Interior Points

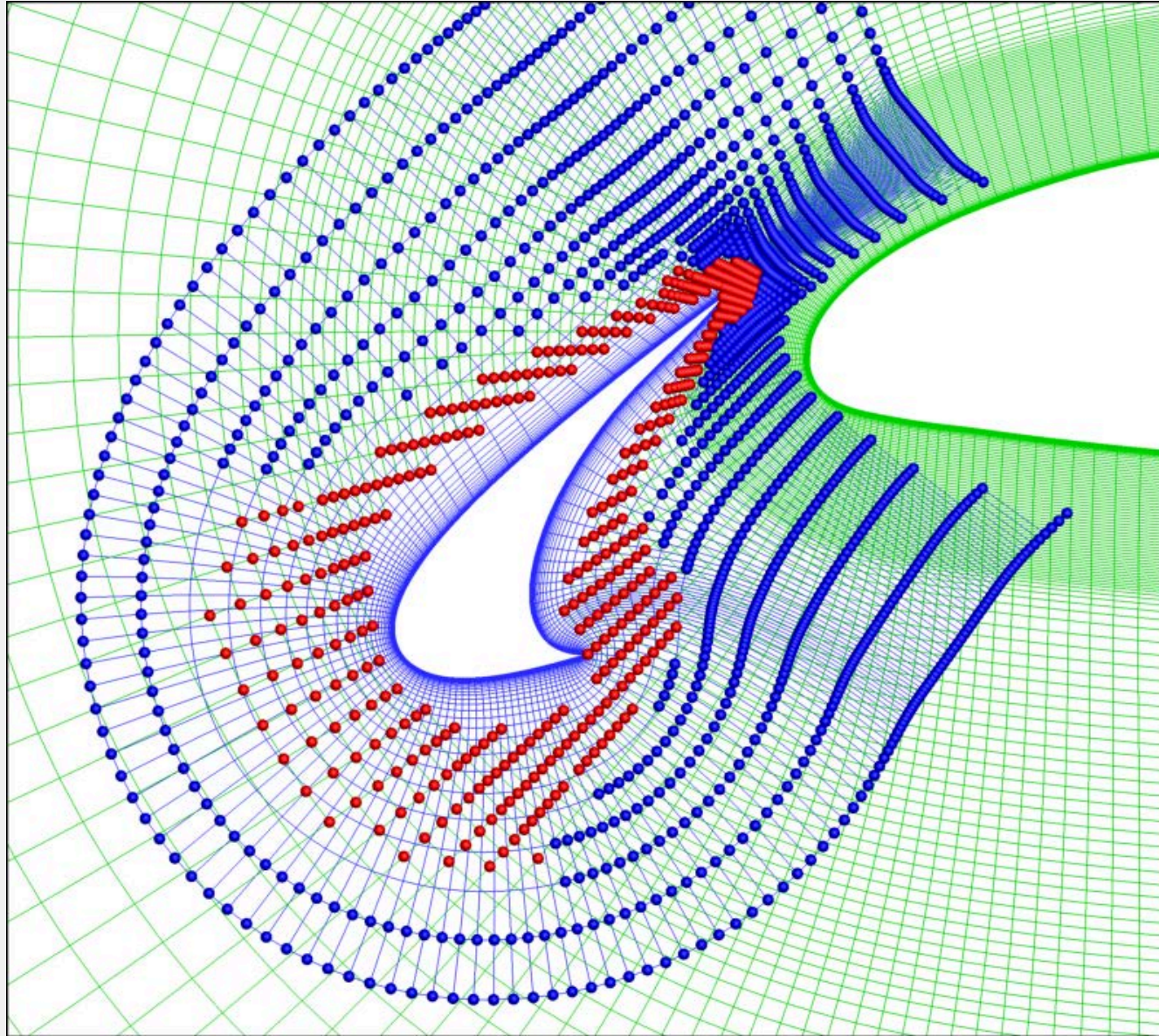


Optimized Overlap Example: Multi-Element Airfoil



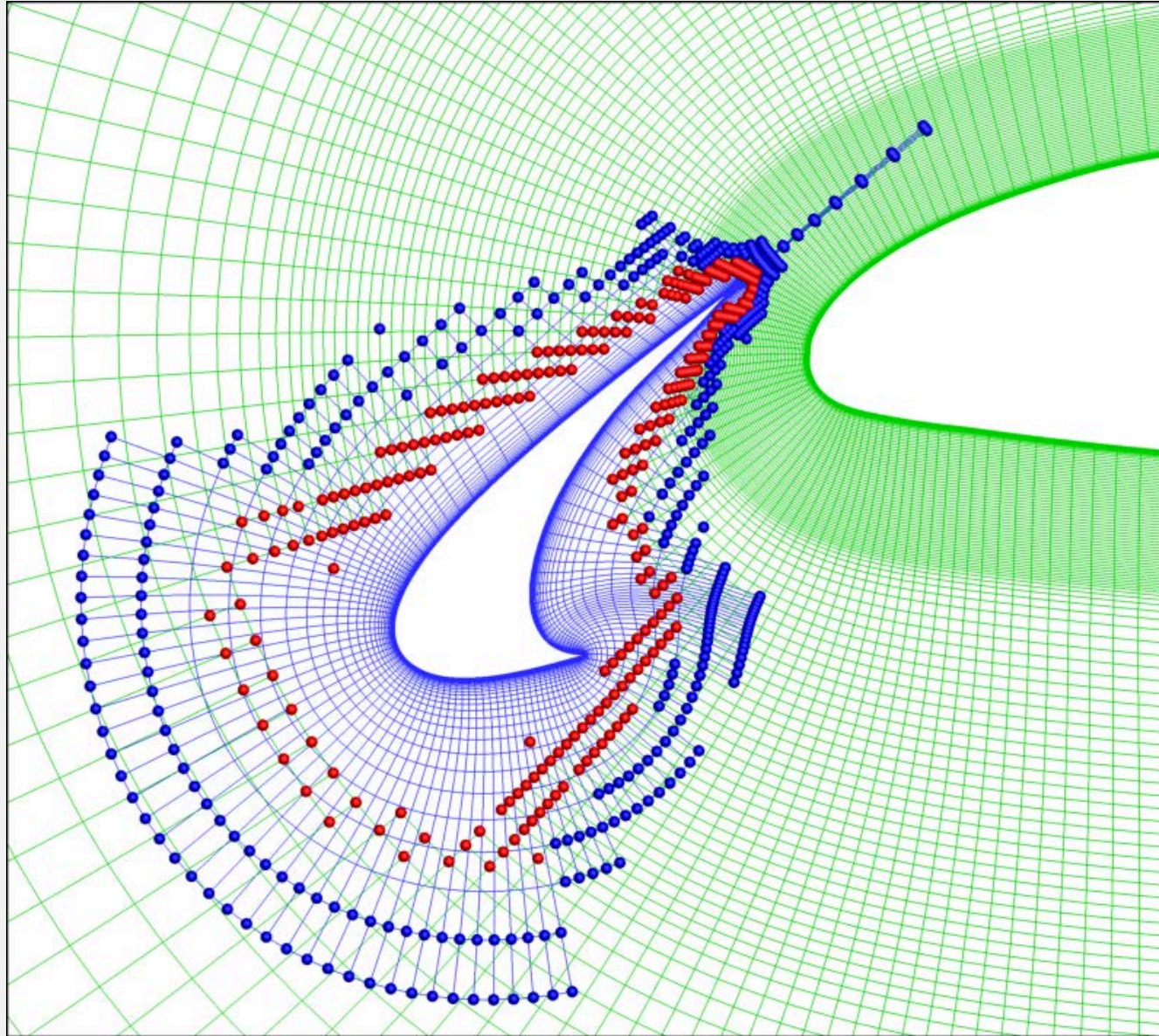
Level 1 Fringes

Optimized Overlap Example: Multi-Element Airfoil



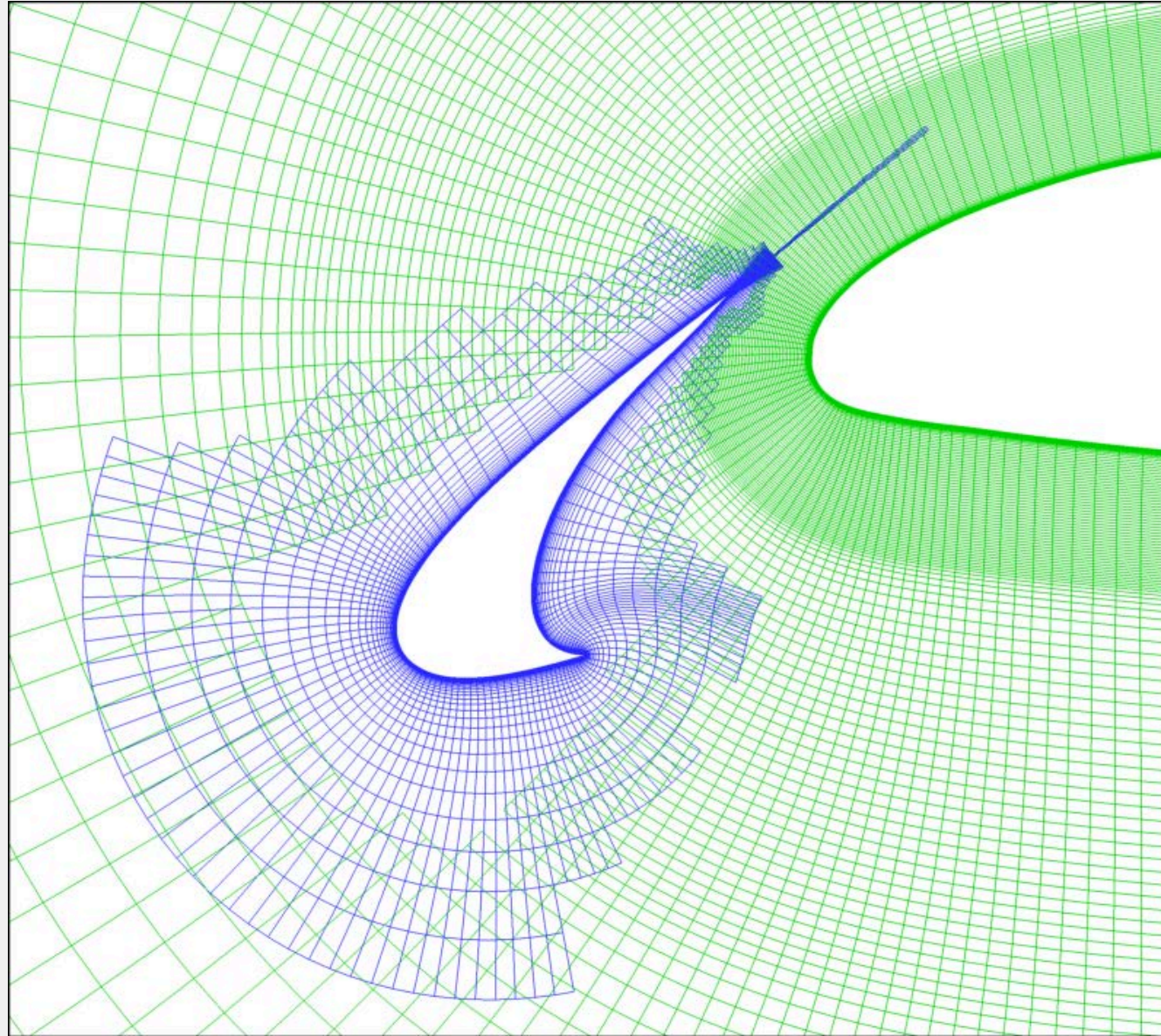
Level 2 Fringes

Optimized Overlap Example: Multi-Element Airfoil



Virtual Fringes

Optimized Overlap Example: Multi-Element Airfoil



Virtual Overlap

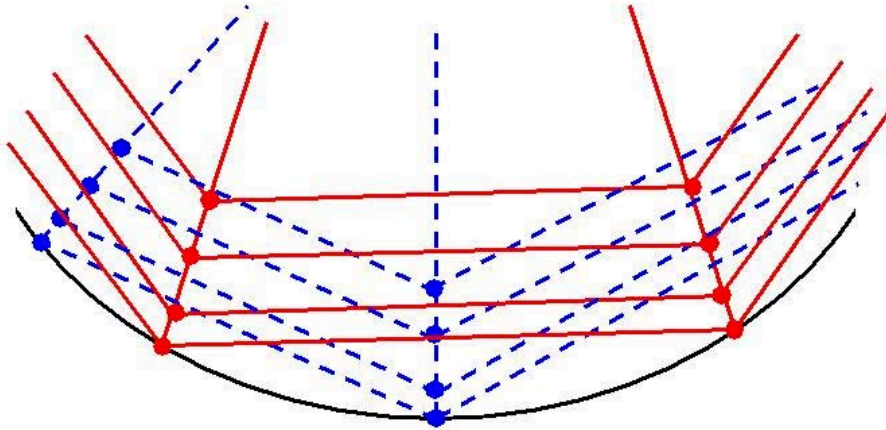


Projection

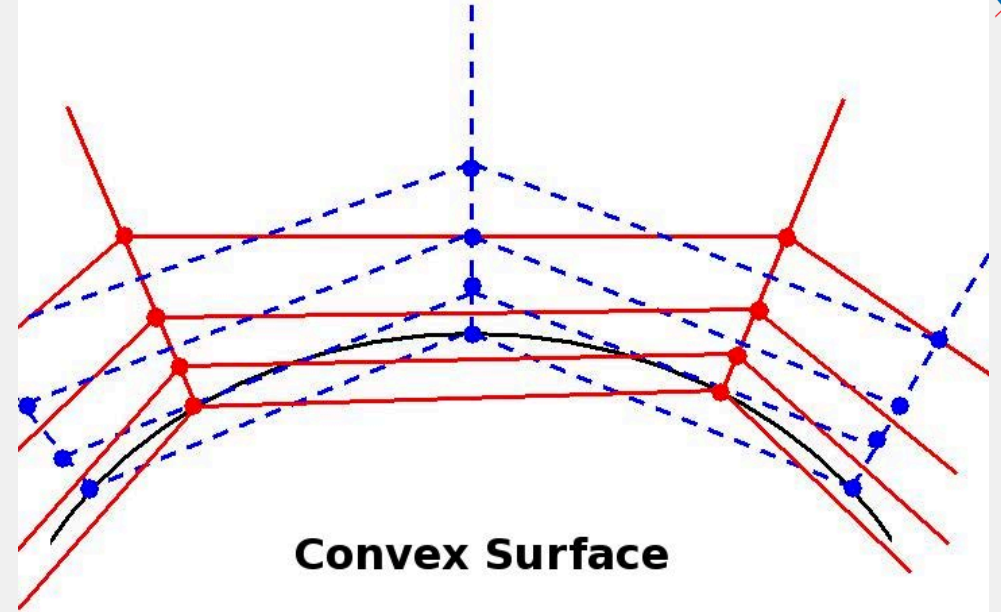
- Corrects interpolation problems that may occur on overlapping curved viscous surfaces
- Cell-aspect ratio typically > 1000 near viscous walls
- Pegasus5 projection step alters interpolation coefficients, not actual grid points
- Projection is performed internally and typically requires no user input

Problem: Linear Discretization on Curved Surfaces

Concave Surface

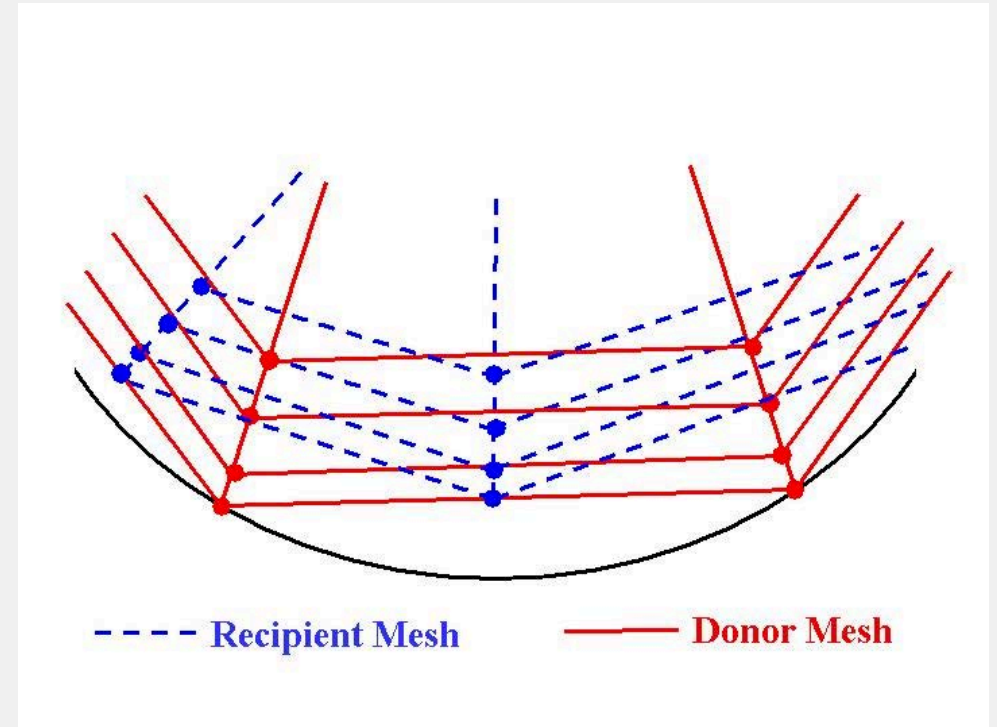
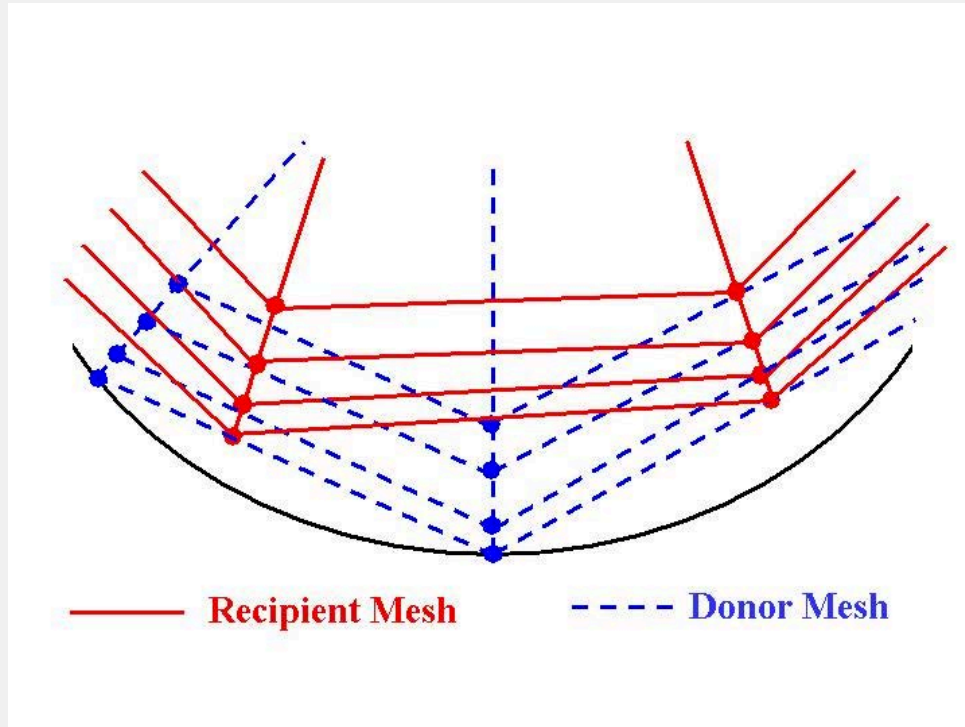


Convex Surface



Solution: Projection

Points are Projected for Interpolation Only

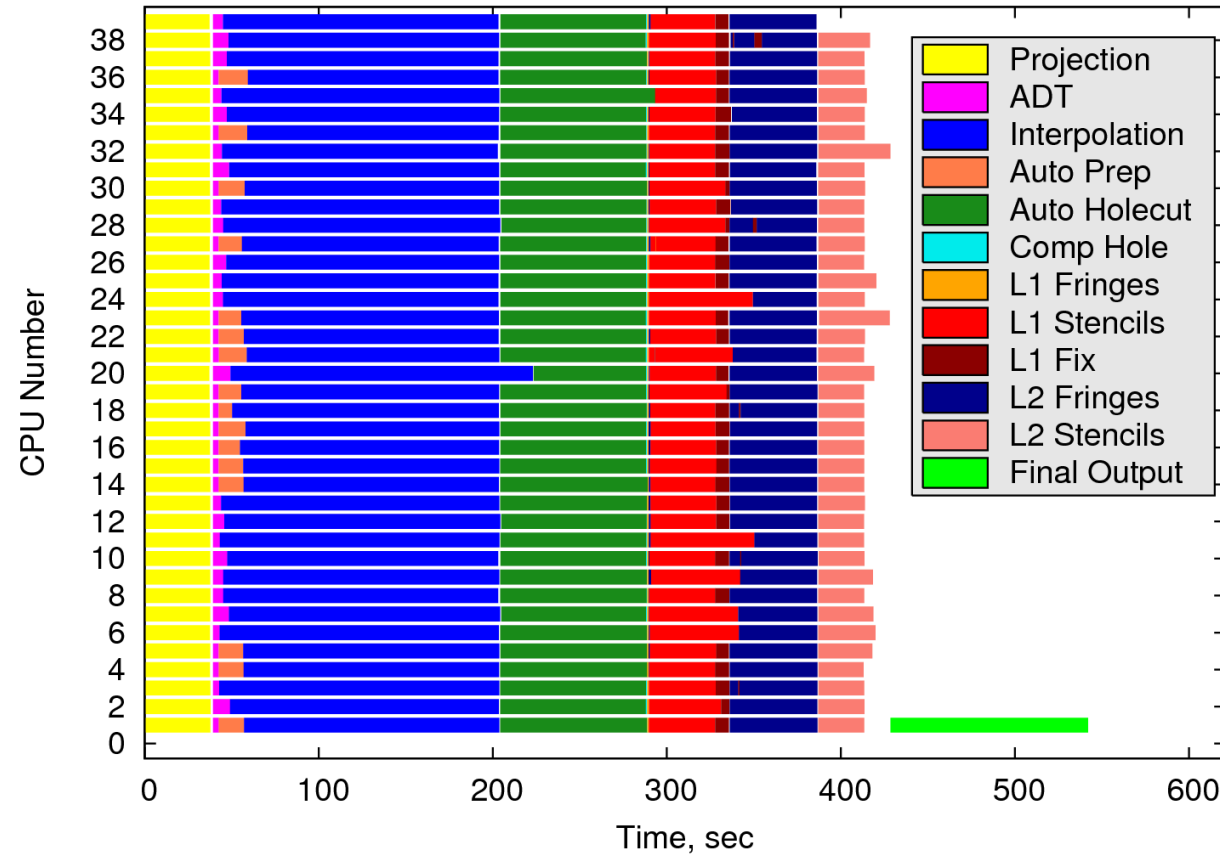


Parallelization

- 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

Parallel Performance

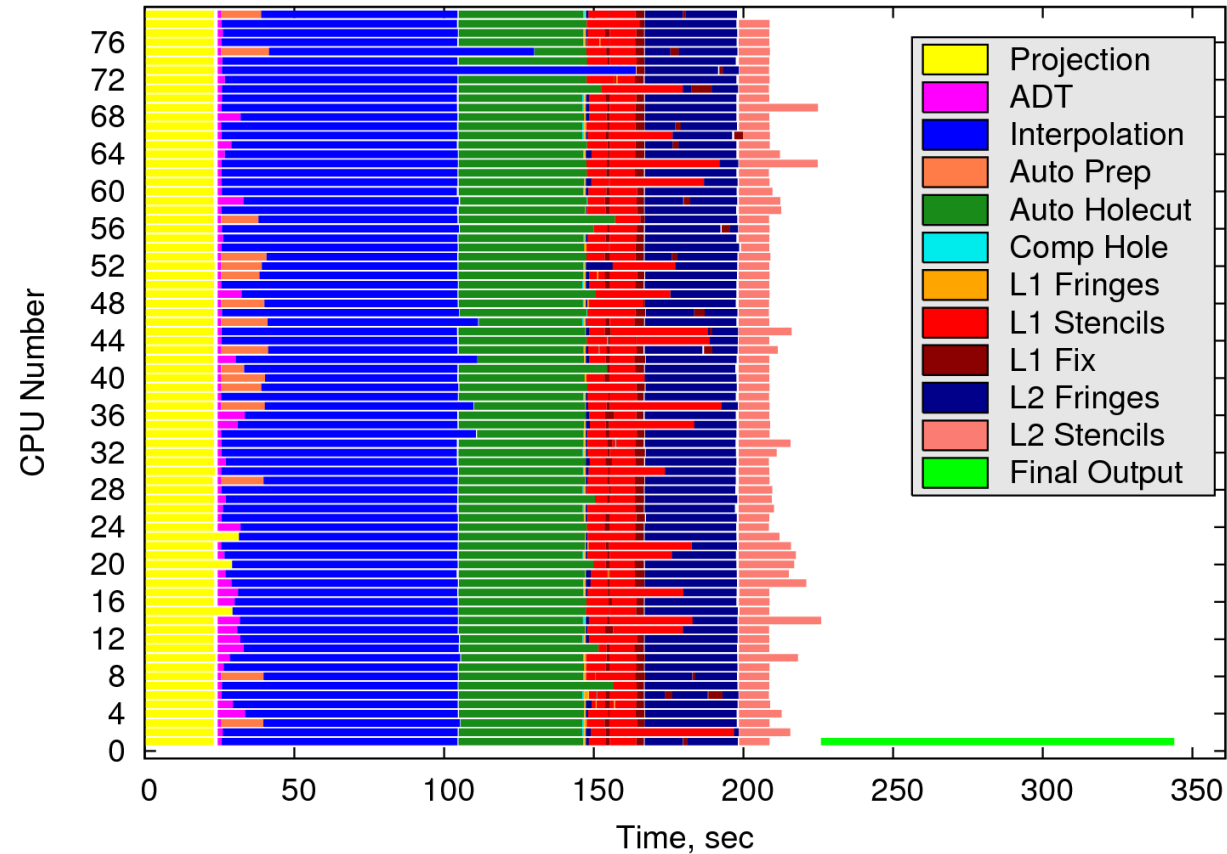
Space Launch System Vehicle: 892 zones, 375 million points



40 Cores: 544 seconds

Parallel Performance

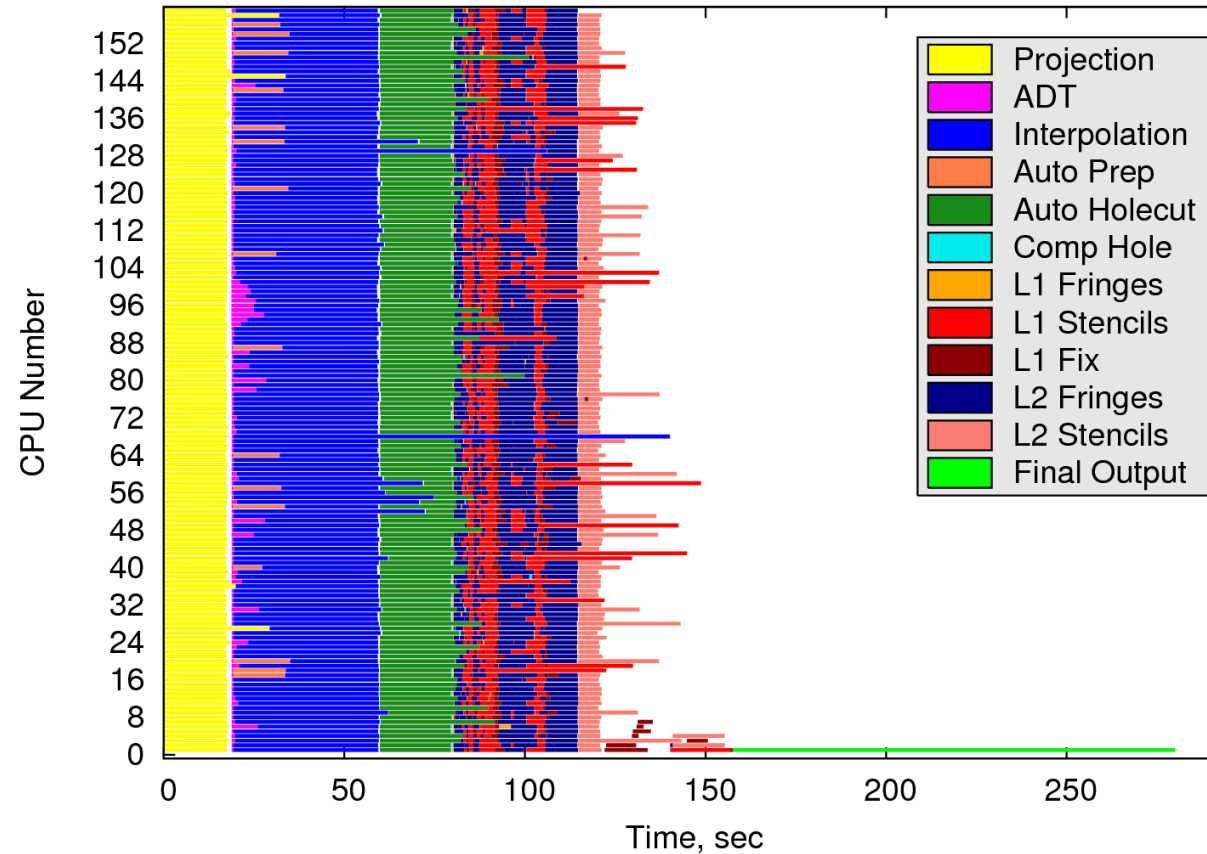
Space Launch System Vehicle: 892 zones, 375 million points



80 Cores: 349 sec

Parallel Performance

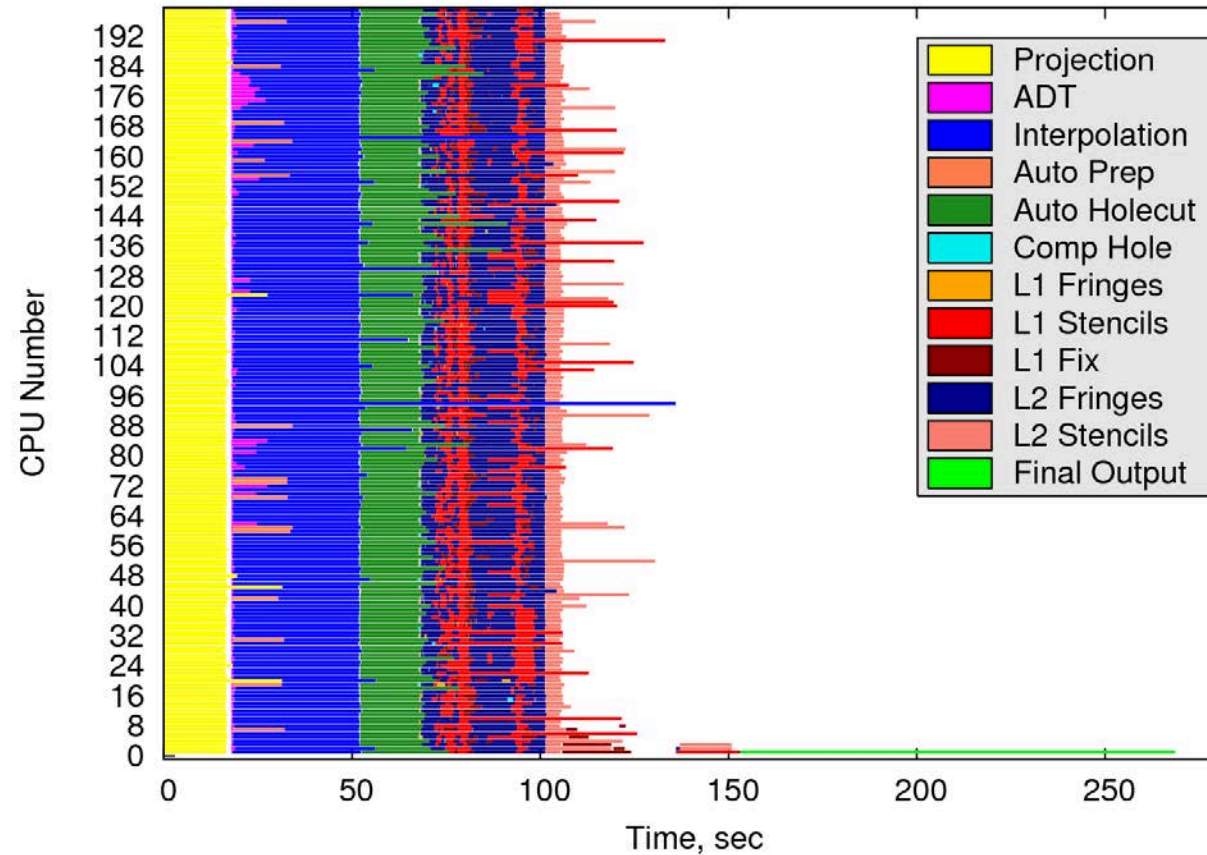
Space Launch System Vehicle: 892 zones, 375 million points



160 Cores: 285 sec

Parallel Performance

Space Launch System Vehicle: 892 zones, 375 million points



200 Cores: 277 sec

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

1. *Introduction and Background*
2. *Overset-grid workflow*
3. *Pegasus5 features and automation*
4. **Overview of usage**
 - Required input
 - Basic usage
 - Understanding the output
 - HCUT and AHCUT
 - Overcoming problems
5. Conclusion

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 an Overflow input file and multi-zone grid file
 - Chimera Grid Tools scripts: BuildPeg5i
 - overgrid

Pegasus5 Input File Example: Bump on Hemisphere Cylinder

```

$GLOBAL
  FRINGE = 2,
$END

$MESH NAME =      'body', LINCLUDE=  2, -1, $END

$MESH NAME = 'bodycap1', LINCLUDE=  2, -1, $END

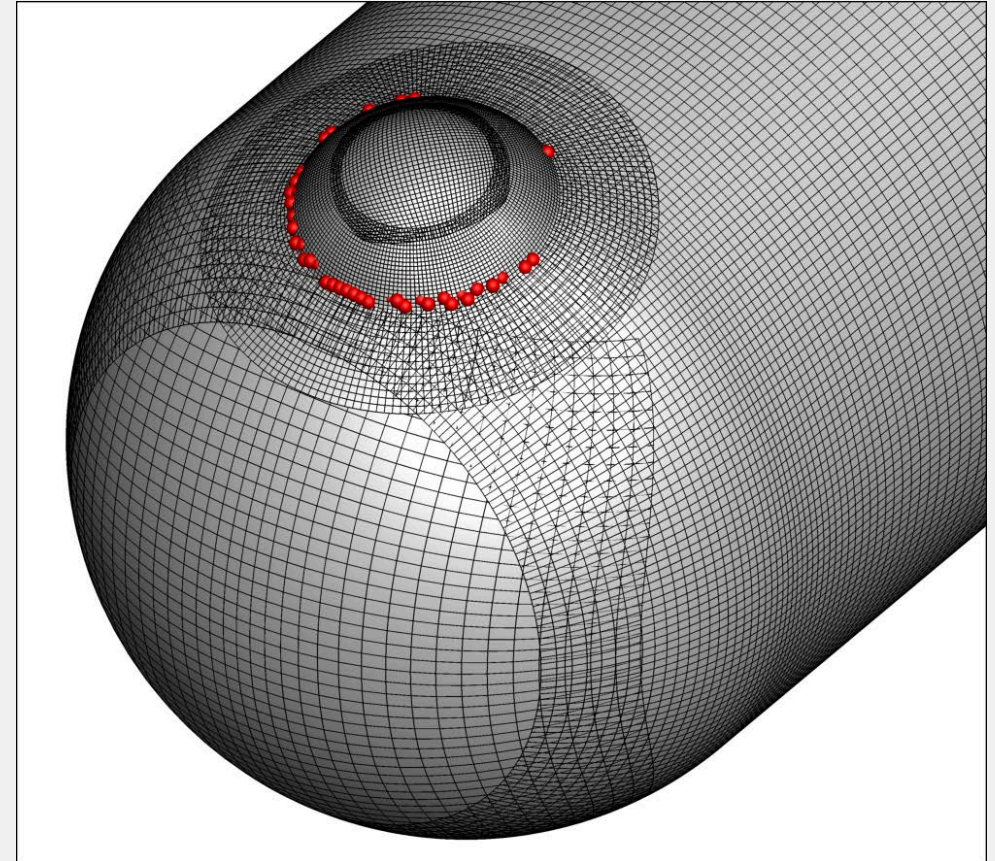
$MESH NAME = 'bodycap2', LINCLUDE=  2, -1, $END

$MESH NAME = 'bumpcap', LINCLUDE=  2, -1, $END

$MESH NAME = 'bumpcol', LINCLUDE=  2, -1, $END

$BCINP ISPARTOF =      'body',
  IBTYP =      5,      47,      10,
  IBDIR =      3,      -3,      2,
  JBCS  =      1,      1,      1,
  JBCE  =     -1,     -1,     -1,
  KBCS  =      1,      1,      1,
  KBCE  =     -1,     -1,      1,

```



Important Boundary Condition Types

- 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 oversight 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 oversight outer boundaries

See Pegasus5 manual for complete list

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
```

```
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, which 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: `rm -rf 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, where XXXX MPI process number
- 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 Boundary Points	Interpolation Stencil	Orphan Points	Hole Points
body	Level 1: 14089 Level 2: 28125 Total: 42214	Level 1: 42227 Level 2: 6215 Total: 48442	1st Fringe: 5 2nd Fringe: 0(Fixed) 3rd Fringe: 0(Fixed) Total: 5	10392
wing	Level 1: 74910 Level 2: 42552 Total: 117462	Level 1: 52173 Level 2: 3514 Total: 55687	1st Fringe: 0 2nd Fringe: 0(Fixed) 3rd Fringe: 0(Fixed) Total: 0	0

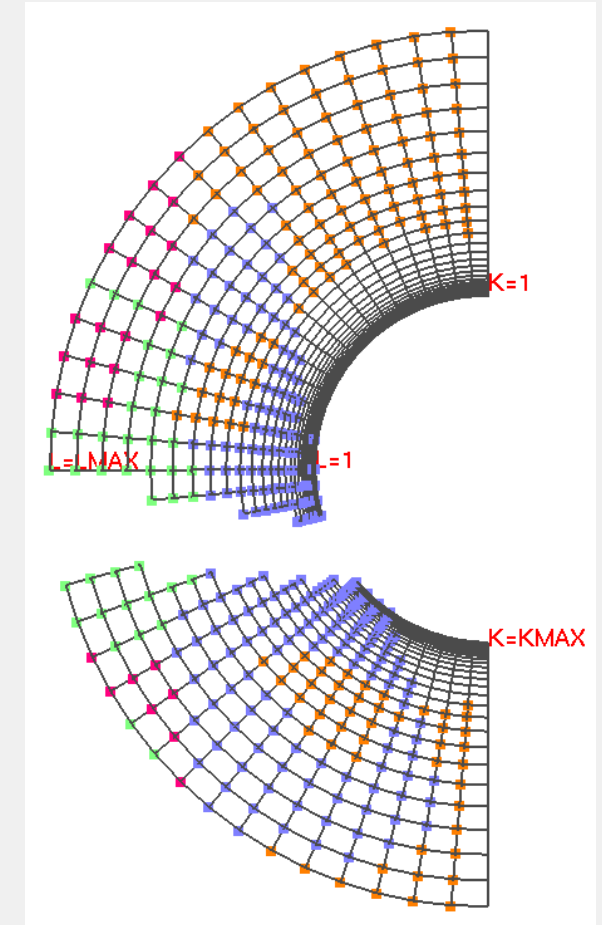
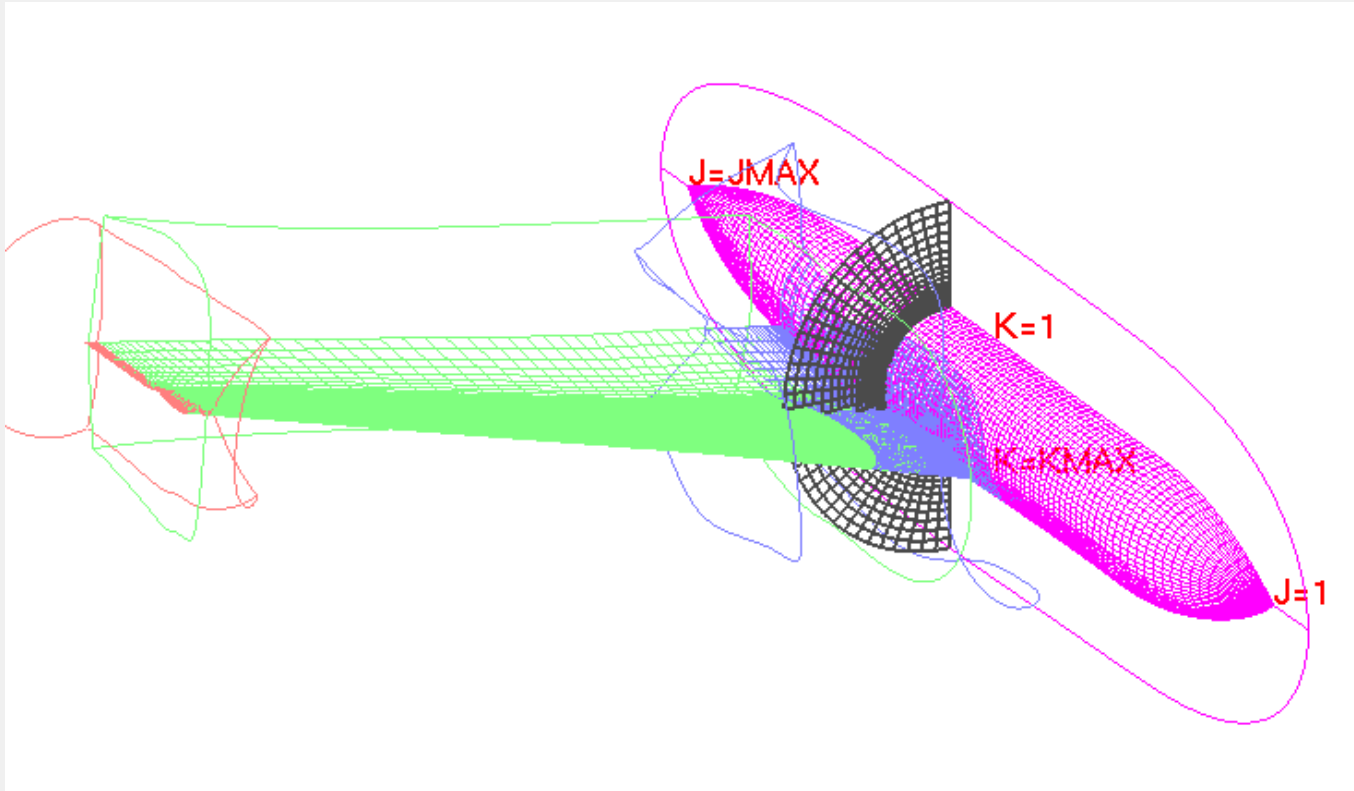
End of log file: Execution Time

PROCESS	CPU(sec)	WALL(sec)	Sub-procs	Max sub-proc(sec)
autohct	0.000	0.000	0	0.000
process_dep	0.254	0.012	31	0.012
projection	1.164	0.207	36	0.207
cellcenter	0.000	0.000	0	0.000
ccbblock1	0.000	0.000	0	0.000
ccbblock2	0.000	0.000	0	0.000
adt	2.695	0.656	7	0.656
auto_hbound	6.516	6.516	1	6.516
man_hbound	0.000	0.000	1	0.000
interpolate	16.832	2.008	36	1.527
auto_cut	9.332	3.594	6	3.594
man_cut	0.090	0.090	1	0.090
comp_hole	0.098	10.027	7	0.031
spec_int1	0.203	10.039	7	0.051
spec_level1	2.188	1.016	7	0.449
level1fix	0.027	0.816	7	0.008
spec_int2	3.504	3.223	18	0.336
spec_level2	1.176	0.852	7	0.242
xintout	0.289	0.289	1	0.289

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 `iblack` array in the grid file, so any `peg_plot` option works when creating the grid file that will be passed to Overflow
 - `XINTOUT` file provides the `iblack` array used by Overflow

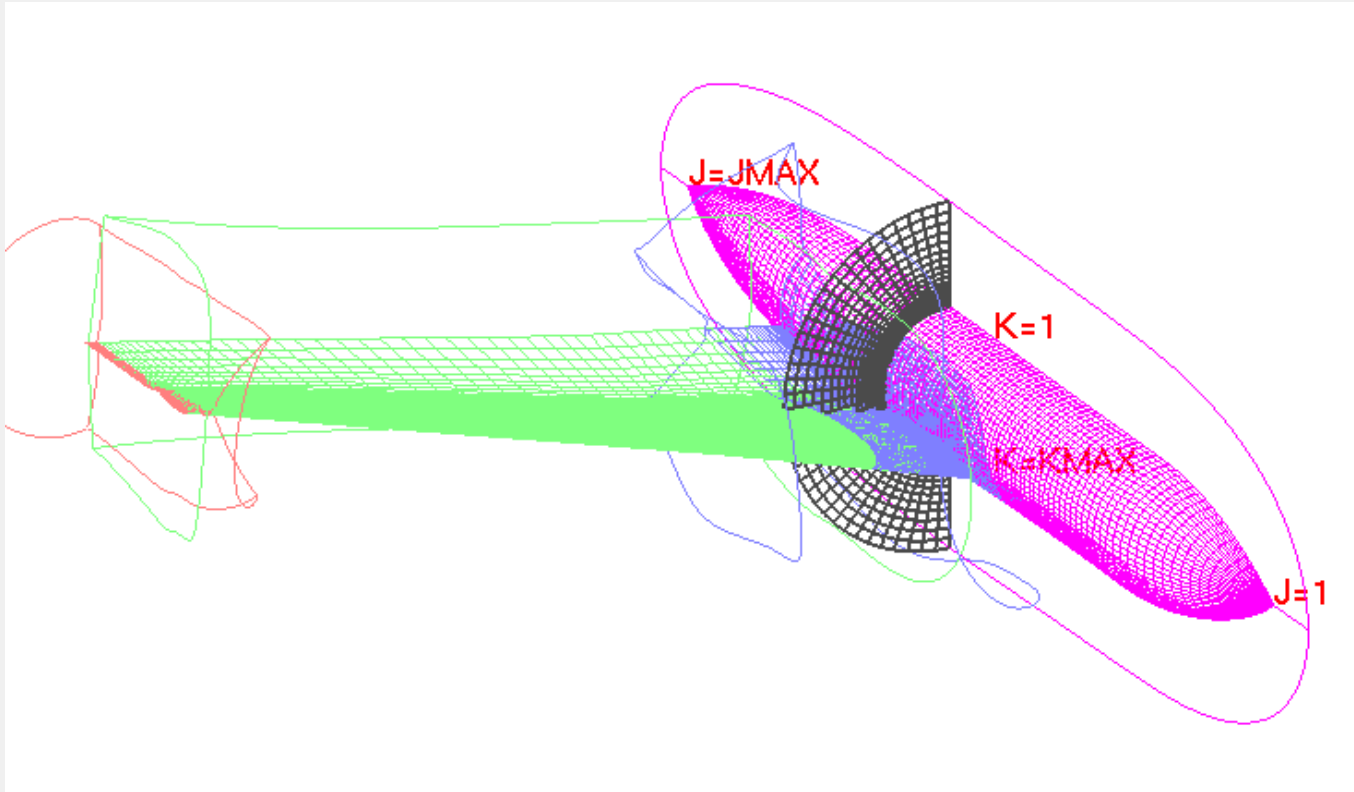
Example: `peg_plot` Option 3



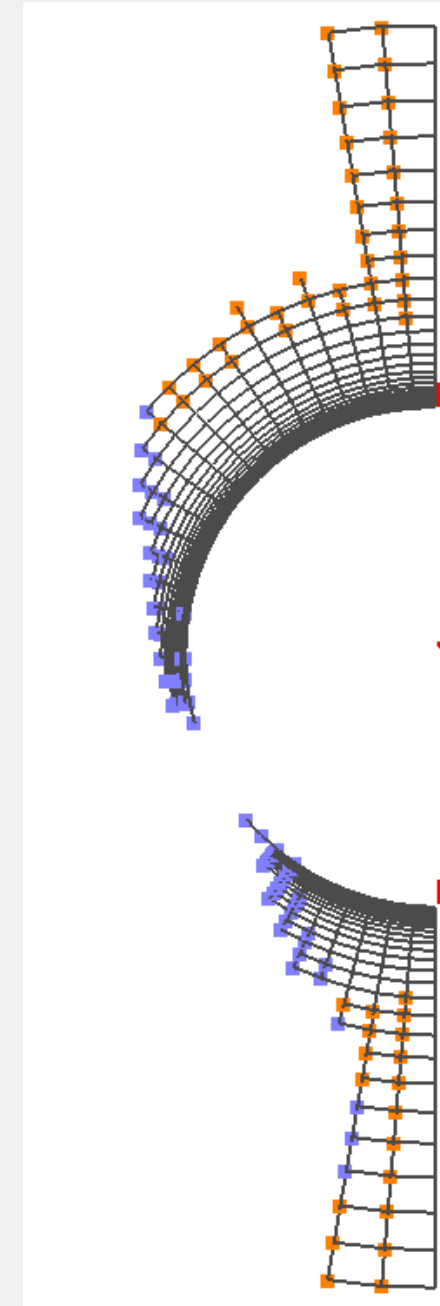
- Wing-body example using `peg_plot` option 3
- View the fuselage zone in overgrid
- Shows auto hole cut by the wing
- Fringe points shown with colored symbols

Example: `peg_plot` Option 2



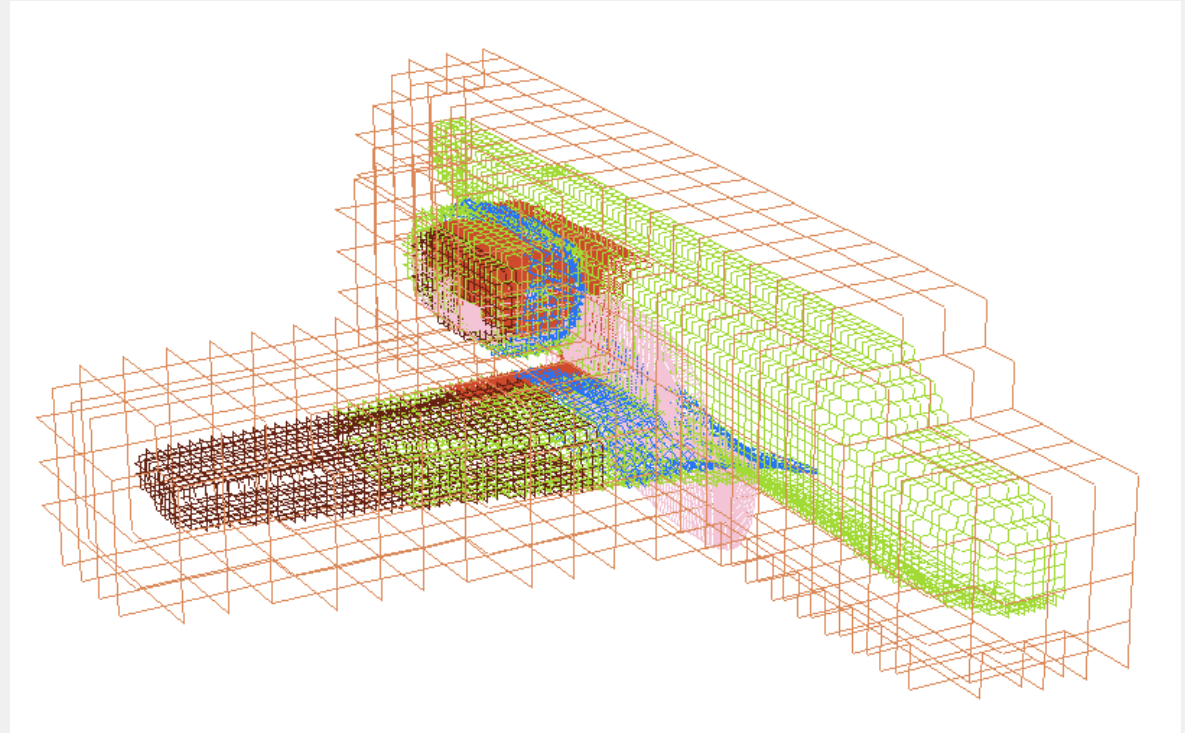


- Wing-body example using `peg_plot` option 2
- Higher-level fringe points have been blanked out
- Shows the virtual overlap after the Level-2 interpolation
- Flow-solver still keeps the higher-level fringes active: they can be used as donor cells for other zones



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



AHCUT and HCUT : Cartesian Hole Cutters

- **HCUT** is basic entity that performs the hole cutting
- **AHCUT** creates one or more **HCUT**s
- **AHCUT** automatically splits a volume into multiple sub-volumes
- **\$HCUT** namelist creates one individual **HCUT** hole-cutter
- **\$AHCUT** namelist creates one or more **HCUT** hole-cutters
- **\$CUTTER_CONTROL** namelist sets defaults for all **\$AHCUT** and **\$HCUT**

```
$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
```

AHCUT and HCUT : Cartesian Hole Cutters

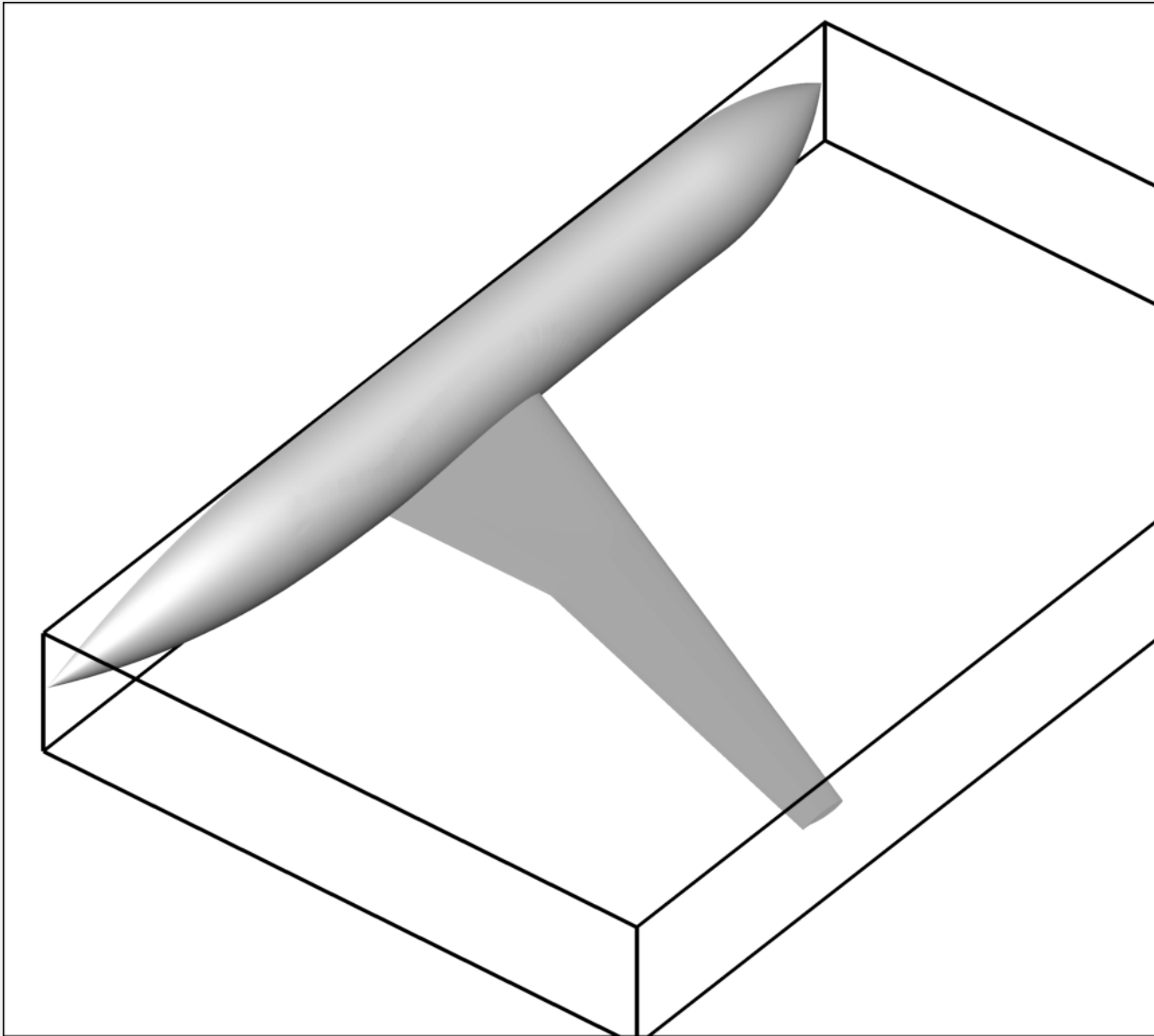
- Default: without any **\$AHCUT** or **\$HCUT** namelists in input file, **pegasus5** uses ALL solid-wall surfaces to cut holes from ALL zones
- Adding either an **\$AHCUT** or **\$HCUT** namelist to input file eliminates the default hole-cutter
- Use multiple **\$AHCUT** namelist inputs for complex geometries
- Creating multiple **HCUT** cutters increases resolution and parallel efficiency

```
$AHCUT NAME = "hcutter1",  
        NAHCUT = 1,  
        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
```



AHCUT Auto Domain Decomposition

Automatic Creation of HCUT Hole-Cutters To Fit The Geometry



```
$AHCUT NAME = "hcutter1",  
      NAHCUT = 1,  
      $END
```



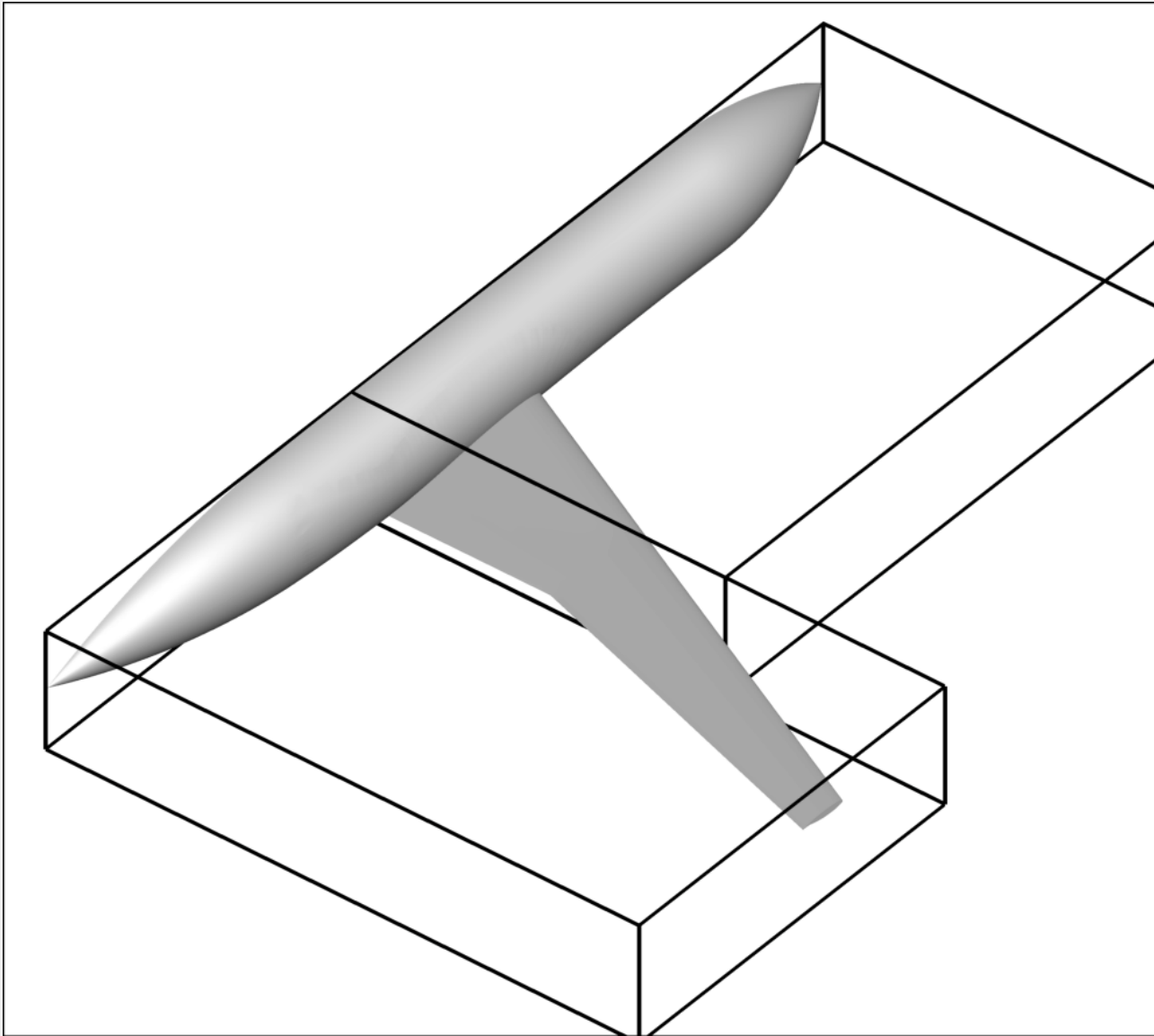


AHCUT Auto Domain Decomposition

Automatic Creation of HCUT Hole-Cutters To Fit The Geometry



```
$AHCUT NAME = "hcutter1",  
    NAHCUT = 2,  
$END
```

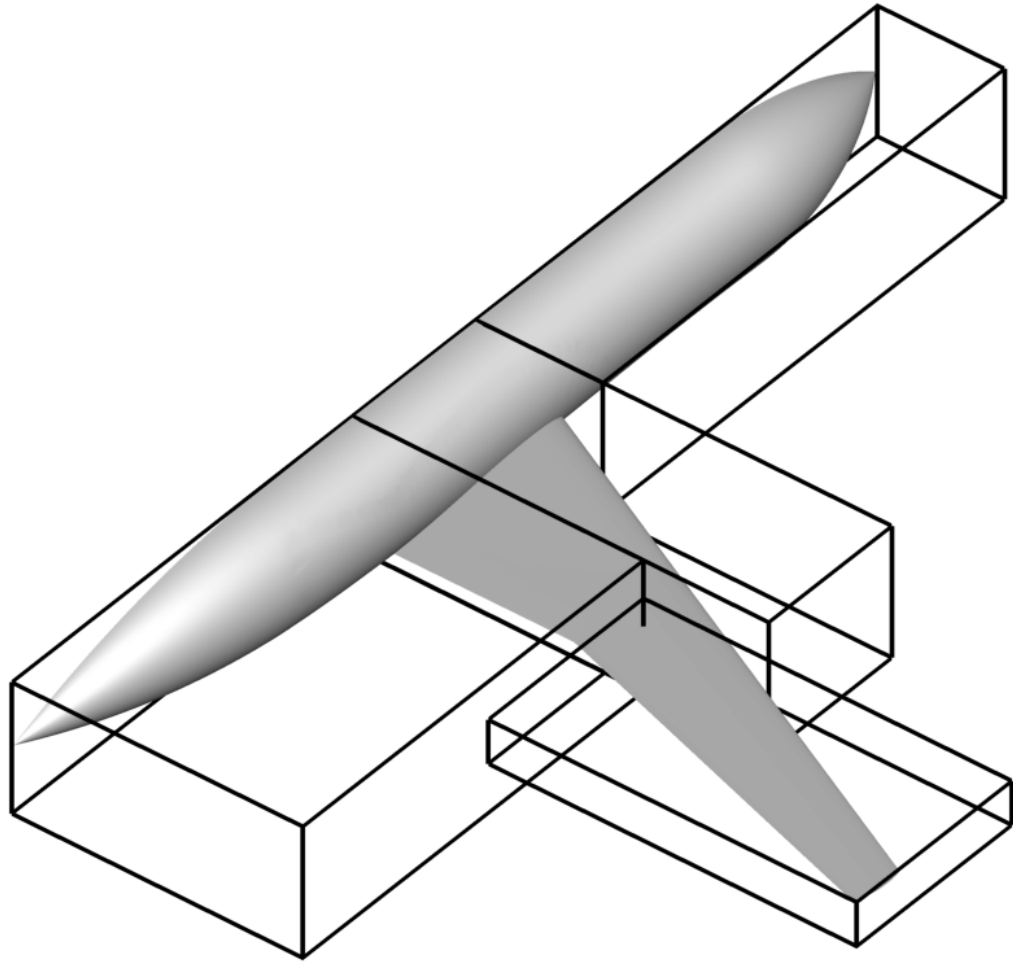




AHCUT Auto Domain Decomposition

Automatic Creation of HCUT Hole-Cutters To Fit The Geometry

```
$AHCUT NAME = "hcutter1",  
    NAHCUT = 4,  
    $END
```



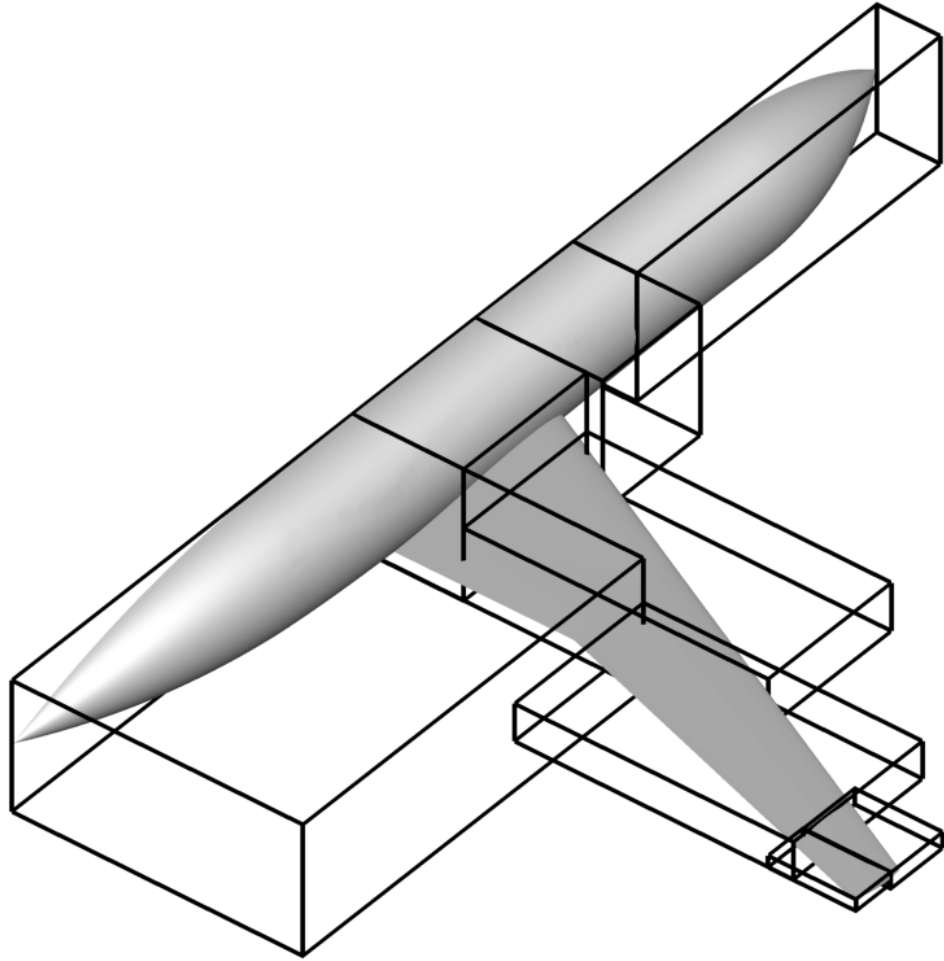


AHCUT Auto Domain Decomposition

Automatic Creation of HCUT Hole-Cutters To Fit The Geometry



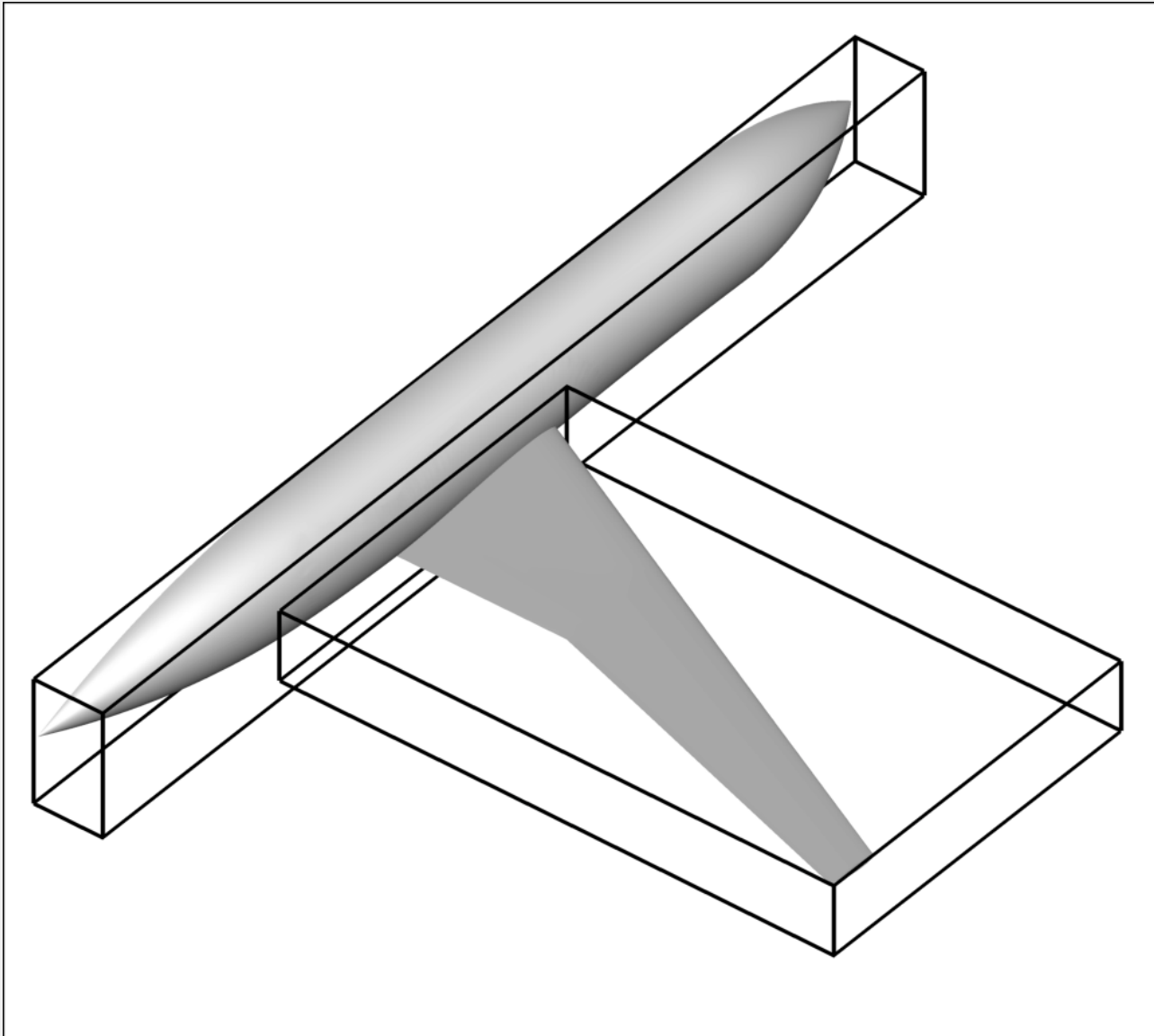
```
$AHCUT NAME = "hcutter1",  
    NAHCUT = 8,  
    $END
```





Control of AHCUT Decomposition

Set first splitting $Y=\text{constant}$ at the wing root



- $CUTPLDIR = \{1, 2, 3\}$
- $CUTPLVAL = \{x, y, z\}$

```
$AHCUT NAME = "hcutter2",  
      NAHCUT = 2,  
      CUTPLDIR = 2,  
      CUTPLVAL = 0.254,  
      $END
```

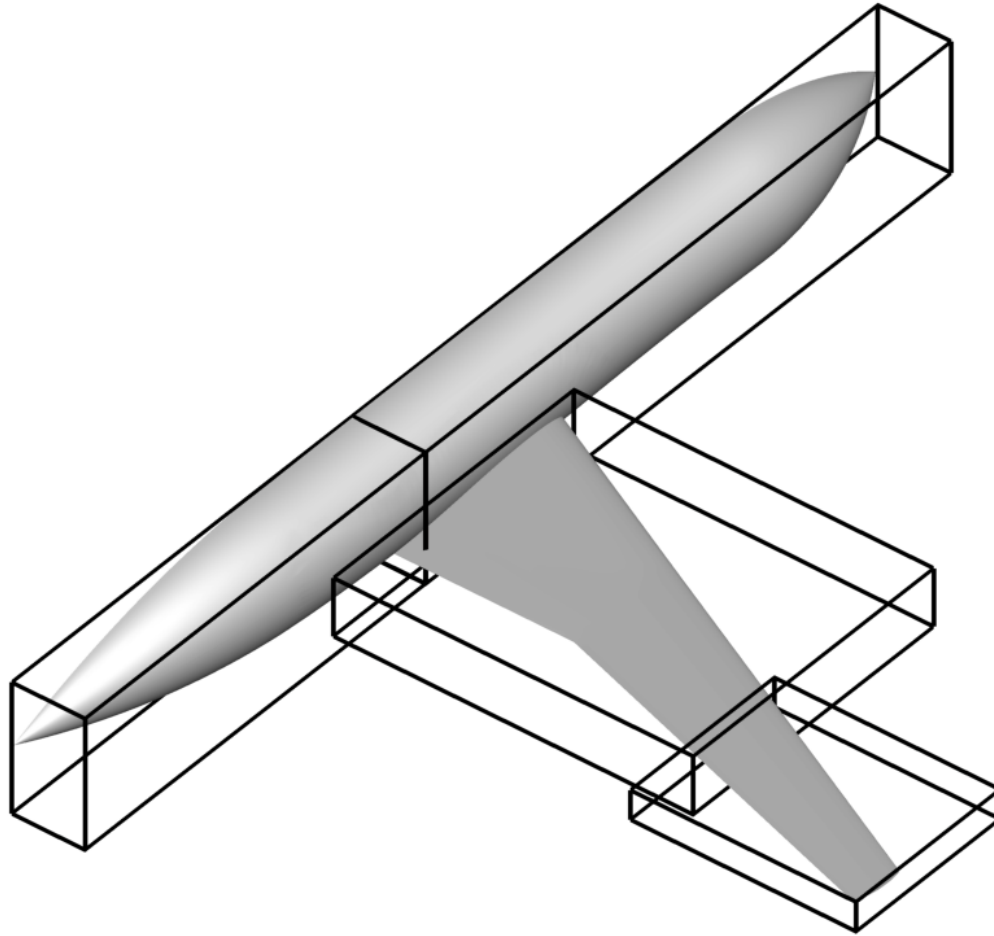


Control of AHCUT Decomposition

Set first splitting $Y=\text{constant}$ at the wing root

- CUTPLDIR={1, 2, 3}
- CUTPLVAL={x, y, z}

```
$AHCUT NAME = "hcutter2",  
      NAHCUT = 4,  
      CUTPLDIR = 2,  
      CUTPLVAL = 0.254,  
      $END
```



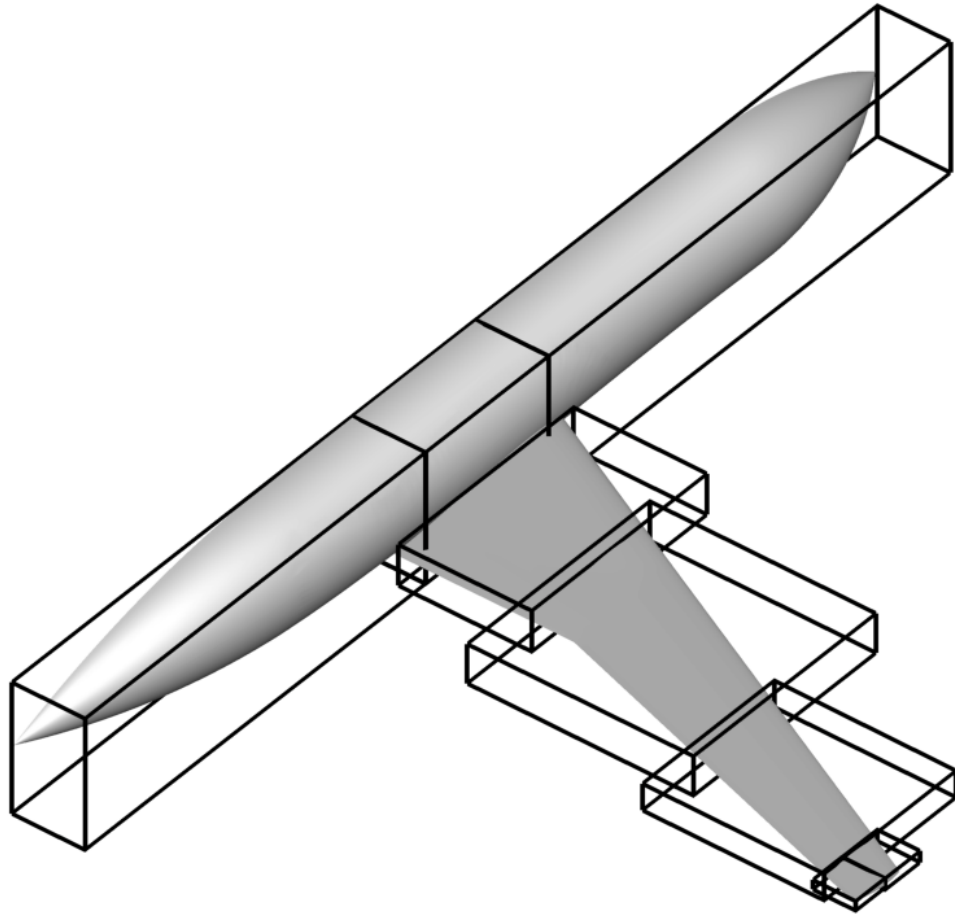


Control of AHCUT Decomposition

Set first splitting $Y=\text{constant}$ at the wing root

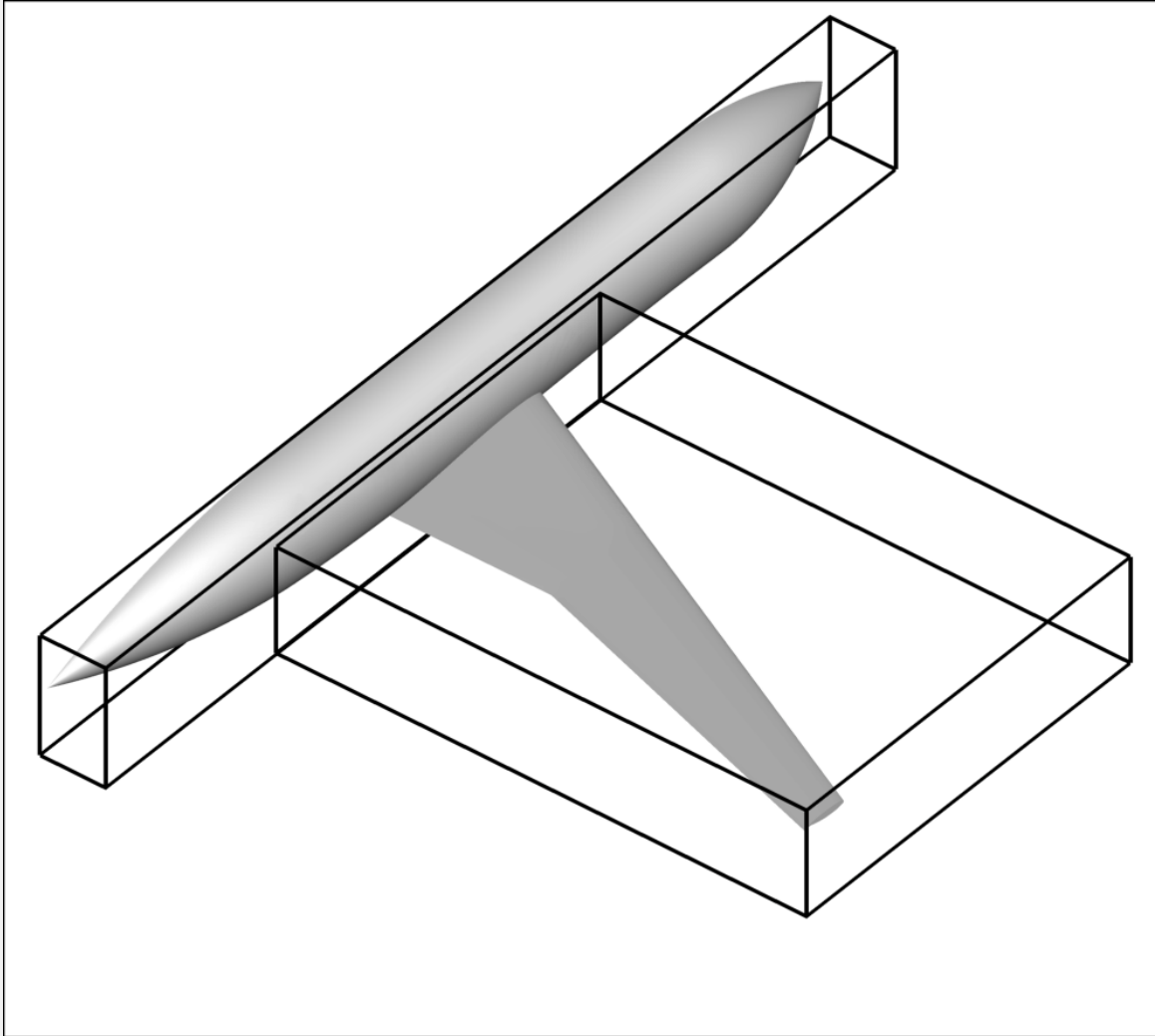
- CUTPLDIR={1, 2, 3}
- CUTPLVAL={x, y, z}

```
$AHCUT NAME = "hcutter2",  
      NAHCUT = 8,  
      CUTPLDIR = 2,  
      CUTPLVAL = 0.254,  
      $END
```



Multiple AHCUT Inputs

Create two separate **AHCUT** inputs



```
$GROUP NAME = 'winggroup',  
  MEMBER = 'wing', 'wingcap', 'wingcol',  
$END
```

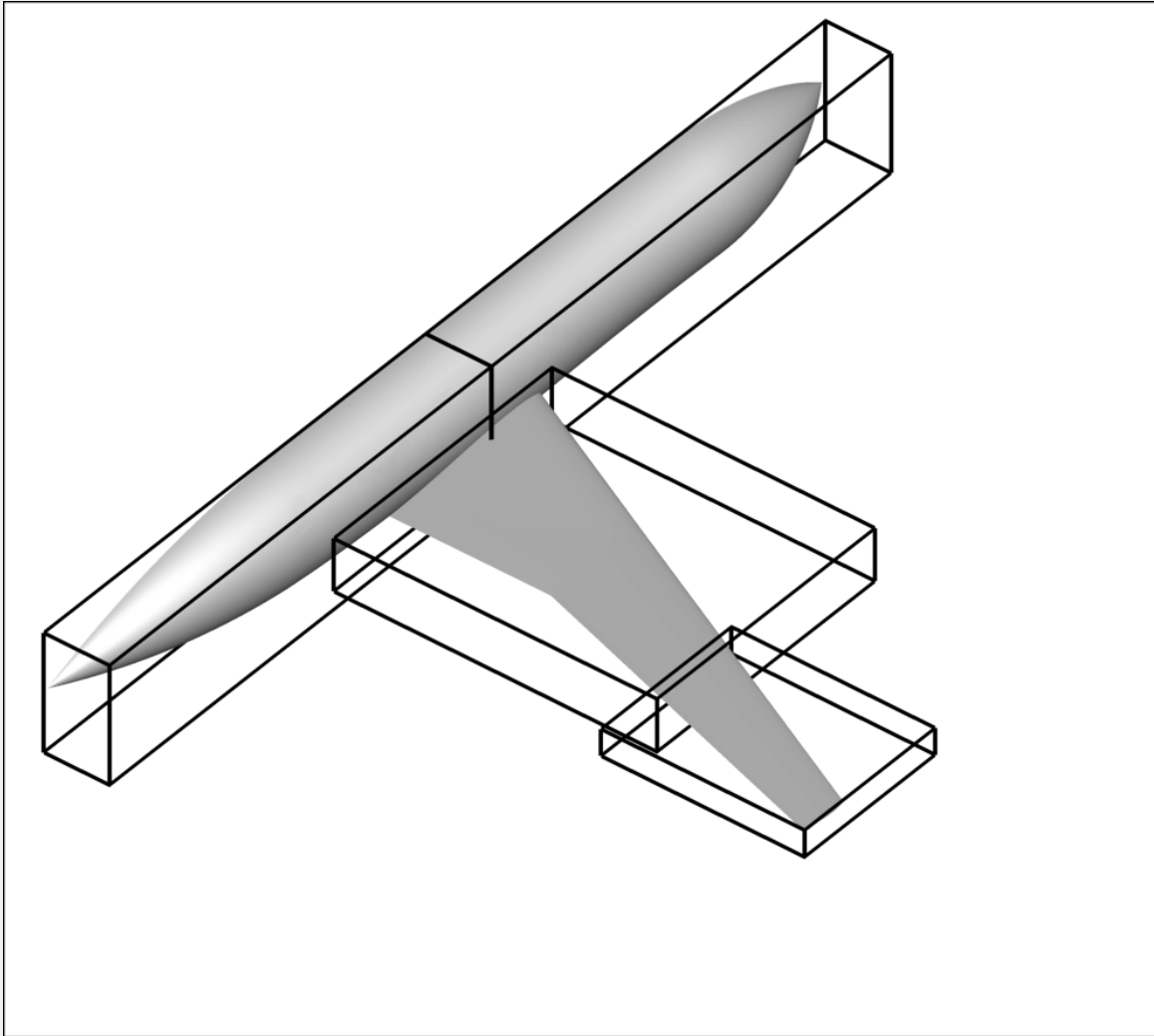
```
$GROUP NAME = 'bodygroup',  
  MEMBER = 'body', 'wingcol',  
$END
```

```
$AHCUT NAME = "bodycutter",  
  NAHCUT = 1,  
  MEMBER = 'bodygroup',  
  CARTY = 0.0, 0.254,  
$END
```

```
$AHCUT NAME = "wingcutter",  
  NAHCUT = 1,  
  MEMBER = 'winggroup',  
  CARTY = 0.254, 2.5,  
$END
```

Multiple AHCUT Inputs

Create two separate **AHCUT** inputs



```
$GROUP NAME = 'winggroup',  
  MEMBER = 'wing', 'wingcap', 'wingcol',  
$END
```

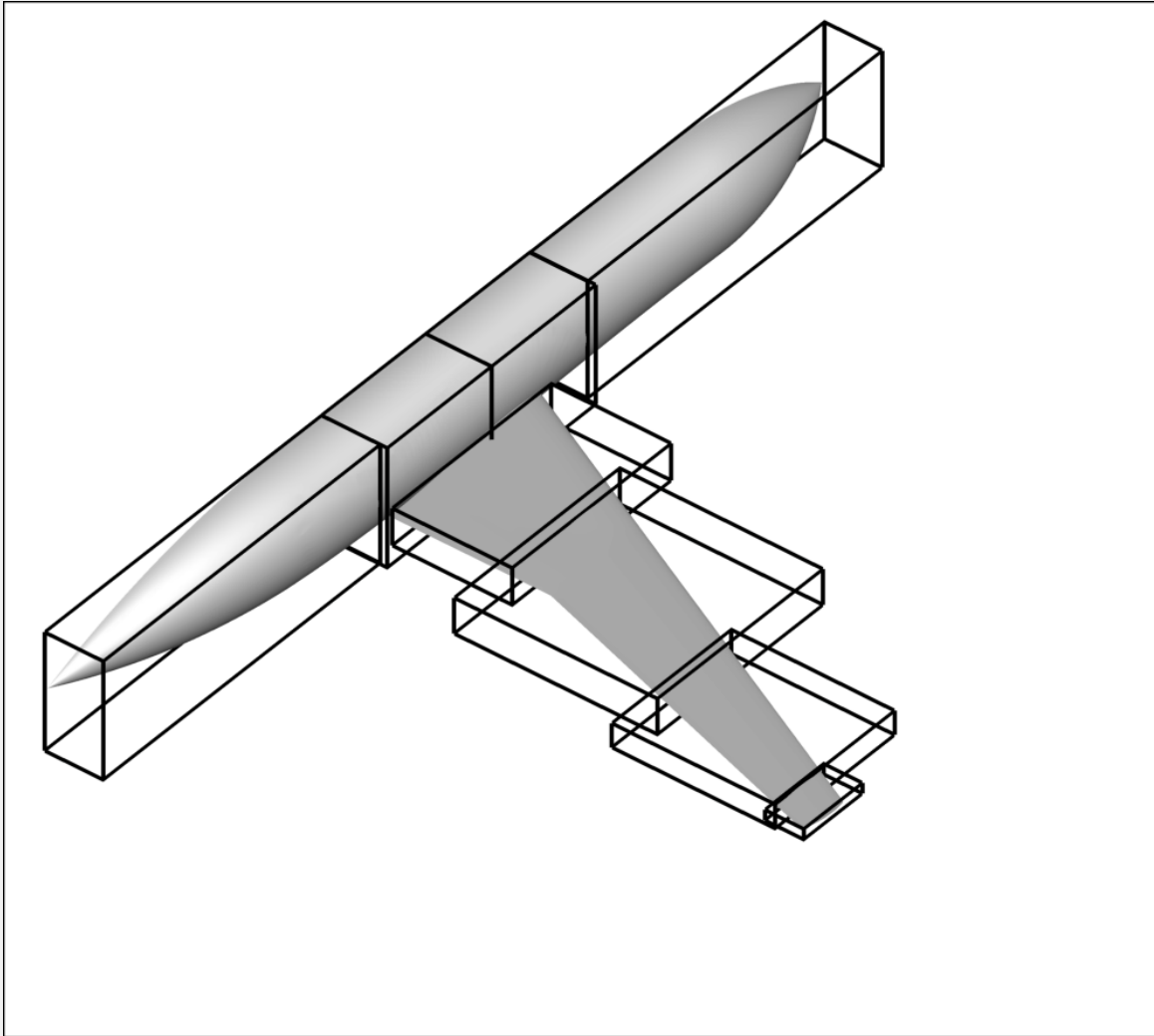
```
$GROUP NAME = 'bodygroup',  
  MEMBER = 'body', 'wingcol',  
$END
```

```
$AHCUT NAME = "bodycutter",  
  NAHCUT = 2,  
  MEMBER = 'bodygroup',  
  CARTY = 0.0, 0.254,  
$END
```

```
$AHCUT NAME = "wingcutter",  
  NAHCUT = 2,  
  MEMBER = 'winggroup',  
  CARTY = 0.254, 2.5,  
$END
```

Multiple AHCUT Inputs

Create two separate **AHCUT** inputs



```
$GROUP NAME = 'winggroup',  
  MEMBER = 'wing', 'wingcap', 'wingcol',  
$END
```

```
$GROUP NAME = 'bodygroup',  
  MEMBER = 'body', 'wingcol',  
$END
```

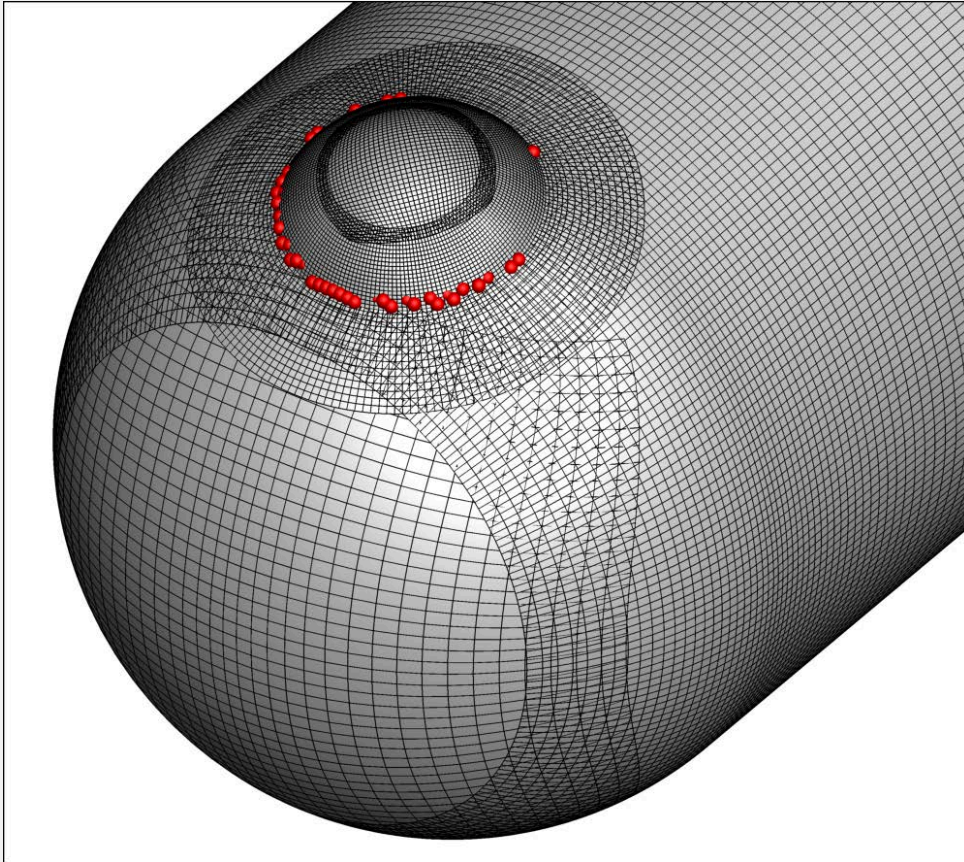
```
$AHCUT NAME = "bodycutter",  
  NAHCUT = 4,  
  MEMBER = 'bodygroup',  
  CARTY = 0.0, 0.254,  
$END
```

```
$AHCUT NAME = "wingcutter",  
  NAHCUT = 4,  
  MEMBER = 'winggroup',  
  CARTY = 0.254, 2.5,  
$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
 - Use **AHCUT**, increased **NAHCUT**
 - Increase OFFSET or HOFFSET to enlarge holes
 - Increase CNX, CNY, CNZ to improve resolution
- Hole points not cut properly near collar grids
 - Use **AHCUT** with increased **NAHCUT**
 - 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

Protuberance HCUT: Bump on Hemisphere Cylinder: Default



```
$GLOBAL
  FRINGE = 2,
$END

$MESH NAME =      'body', LINCLUDE=  2, -1, $END

$MESH NAME = 'bodycap1', LINCLUDE=  2, -1, $END

$MESH NAME = 'bodycap2', LINCLUDE=  2, -1, $END

$MESH NAME = 'bumpcap', LINCLUDE=  2, -1, $END

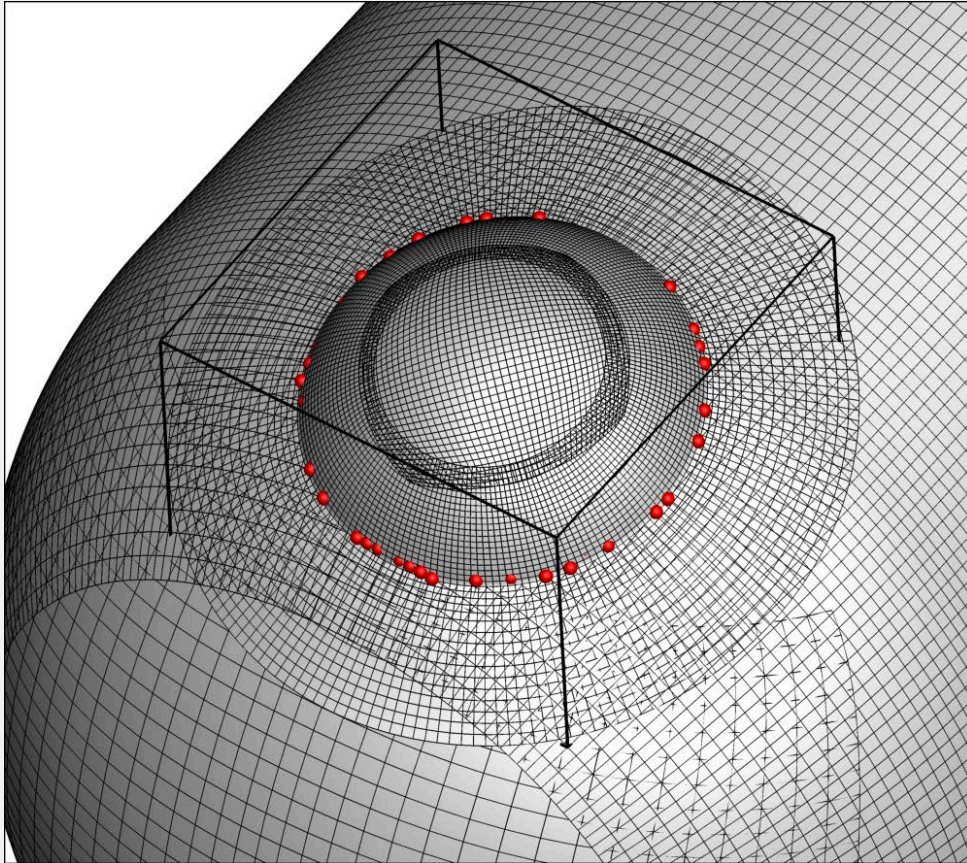
$MESH NAME = 'bumpcol', LINCLUDE=  2, -1, $END

$BCINP ISPARTOF =      'body',
  IBTYP =      5,      47,      10,
  IBDIR =      3,      -3,      2,
  JBCS  =      1,      1,      1,
  JBCE  =     -1,     -1,     -1,
  KBCS  =      1,      1,      1,
  KBCE  =     -1,     -1,      1,
```

Use Default Hole Cutting

Result: 1172 orphans

Protuberance HCUT: Add HCUT Input



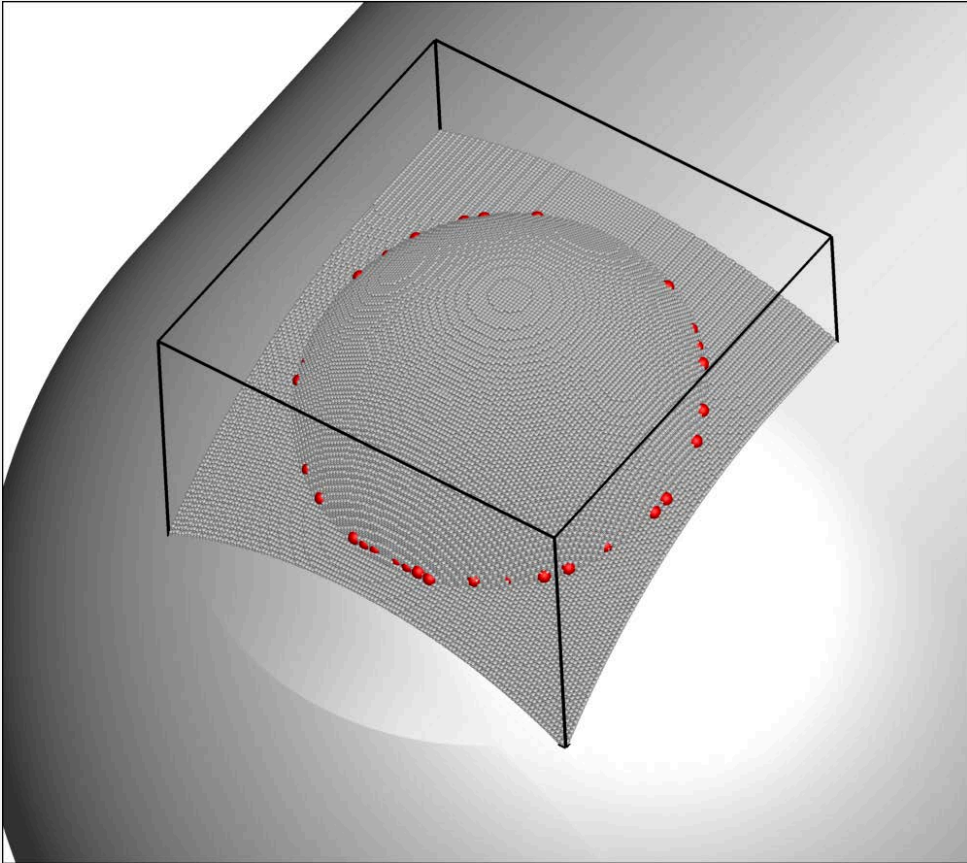
```
$HCUT NAME = 'bump_hcutA',  
  CNX = 128,  
  CNY = 128,  
  CNZ = 128,  
  CARTX = 9.6, 11.6,  
  CARTY = -0.9, 1.1,  
  CARTZ = 1.6, 2.9,  
  CORNERO = 00001111,  
$END
```

Add coarse HCUT

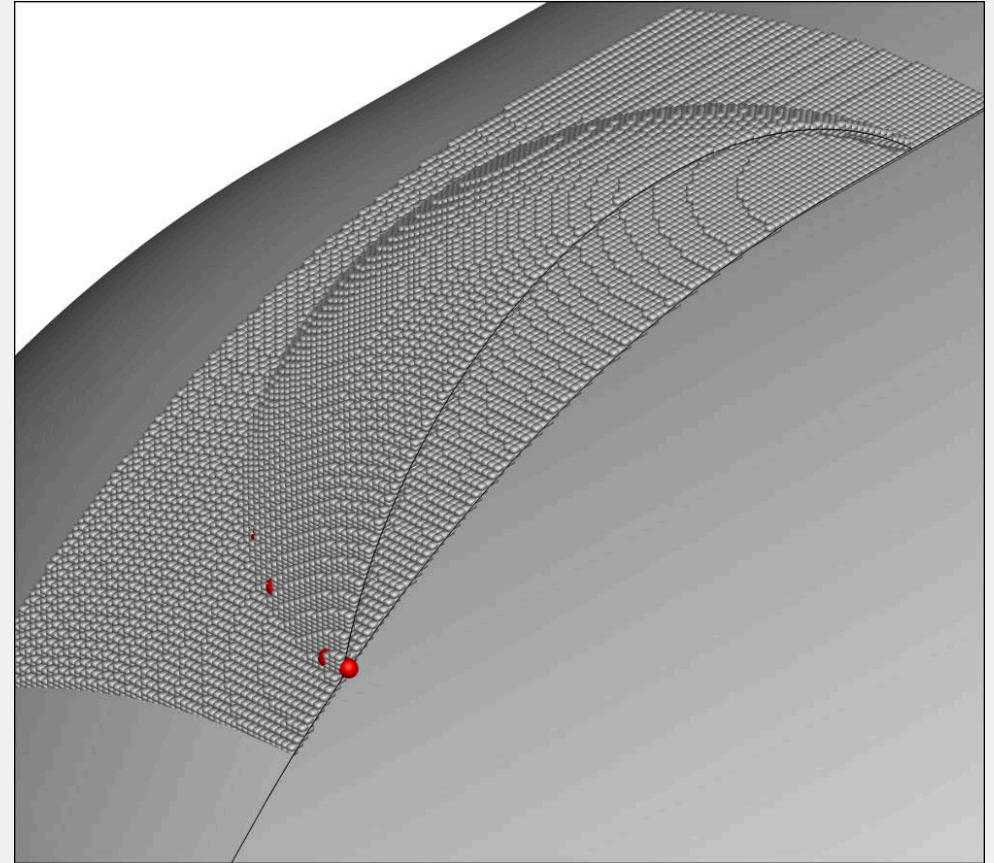
Result: 711 orphans

hcut_plot -edge

Protuberance HCUT: HCUT Fringe Cartesian Cells

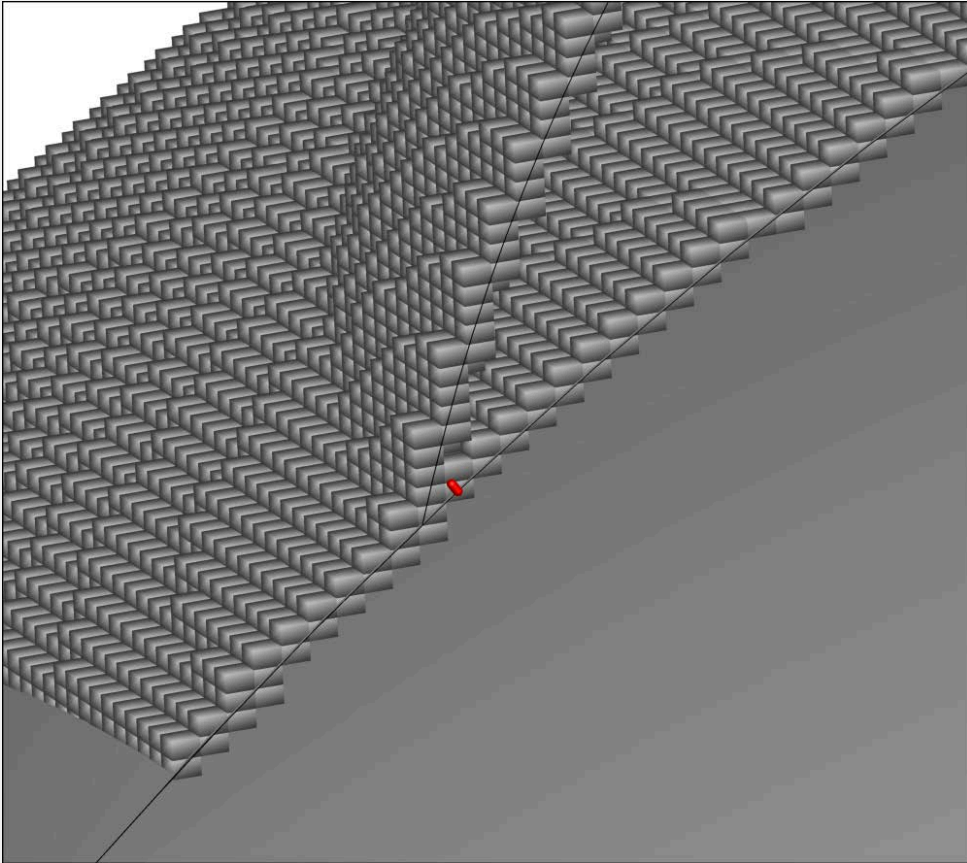


`hcut_plot -fringe`

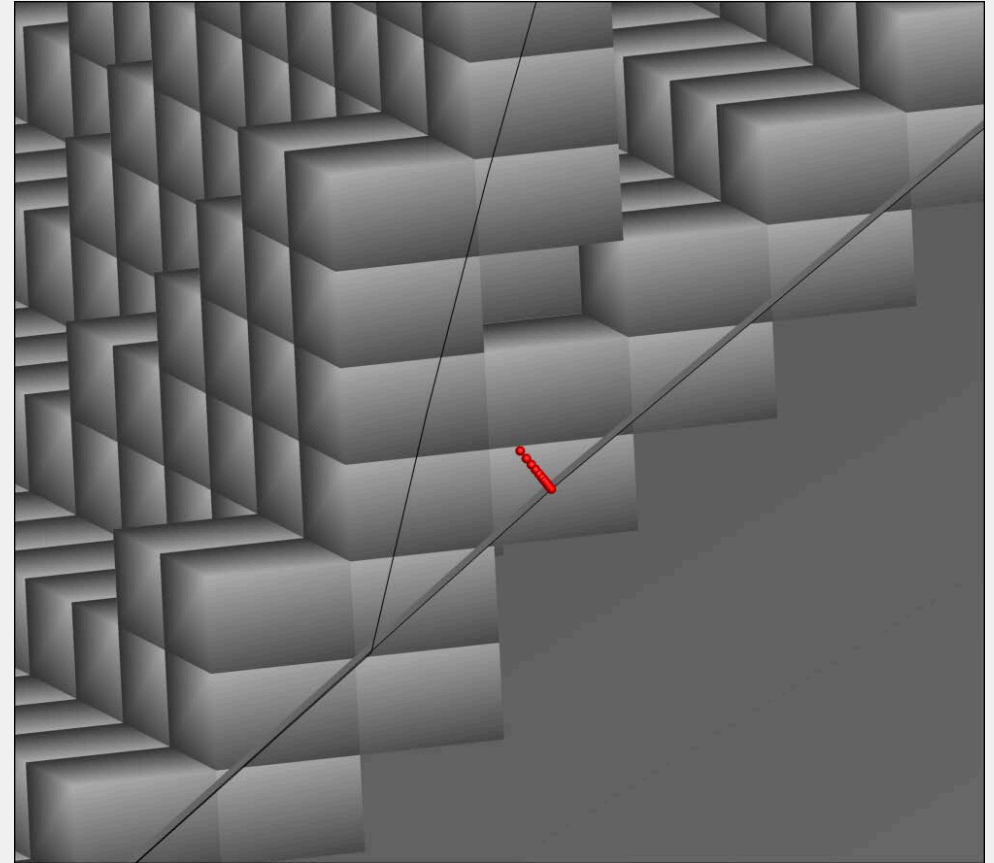


`hcut_plot -fringe`

Protuberance HCUT: HCUT Fringe Cartesian Cells

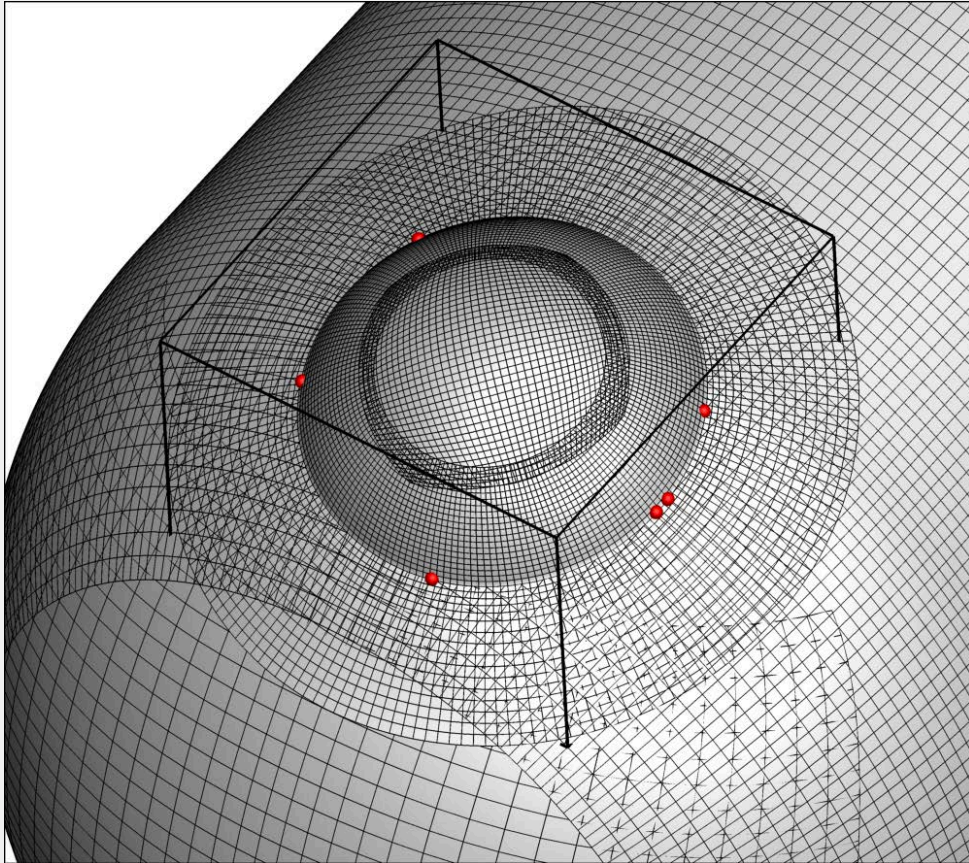


`hcut_plot -fringe`



`hcut_plot -fringe`

Protuberance HCUT: Increase HCUT Resolution



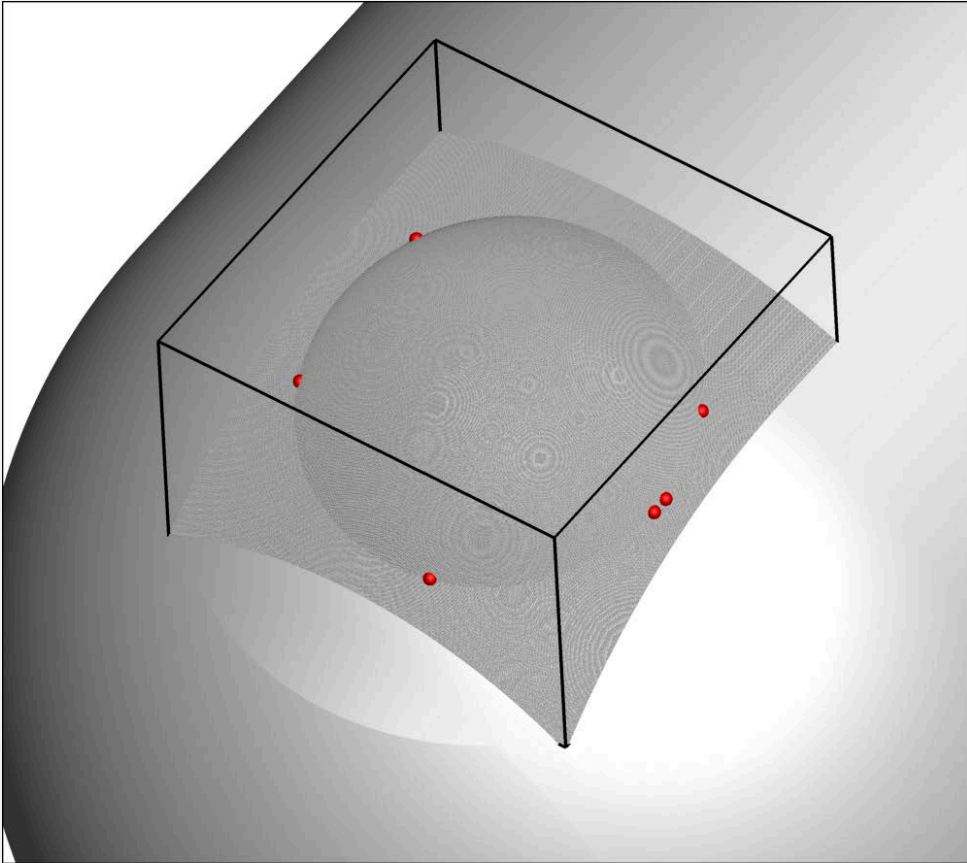
hcut_plot -edge

```
$HCUT NAME = 'bump_hcutB',  
      CNX = 512,  
      CNY = 512,  
      CNZ = 512,  
      CARTX = 9.6, 11.6,  
      CARTY = -0.9, 1.1,  
      CARTZ = 1.6, 2.9,  
      CORNERO = 00001111,  
$END
```

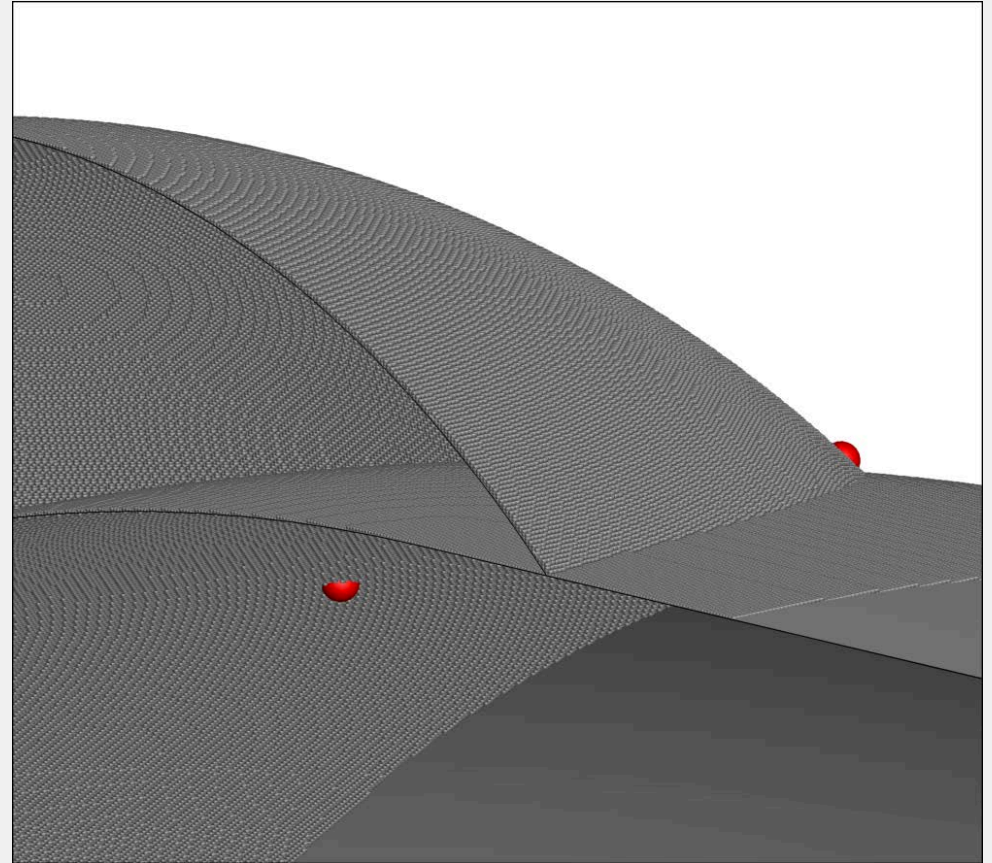
Increase HCUT resolution

Result: 82 orphans

Protuberance HCUT: Increase HCUT Resolution



`hcut_plot -fringe`



`hcut_plot -fringe`

Protuberance HCUT: Use of HOFFSET and OFFSET

```
$MESH NAME = 'body',  
      OFFSET = 1,  
      LINCLUDE = 2, -1,  
$END  
  
$HCUT NAME = 'bump_hcutC',  
      CNX = 512,  
      CNY = 512,  
      CNZ = 512,  
      CARTX = 9.6, 11.6,  
      CARTY = -0.9, 1.1,  
      CARTZ = 1.6, 2.9,  
      CORNERO = 00001111,  
$END
```

Set **\$MESH** input: **OFFSET=1**

Result: 0 orphans

- **OFFSET** grows every hole in that MESH by one cell in each j,k,l direction
- See also **OFFSTJ**, **OFFSTK**, **OFFSTL**

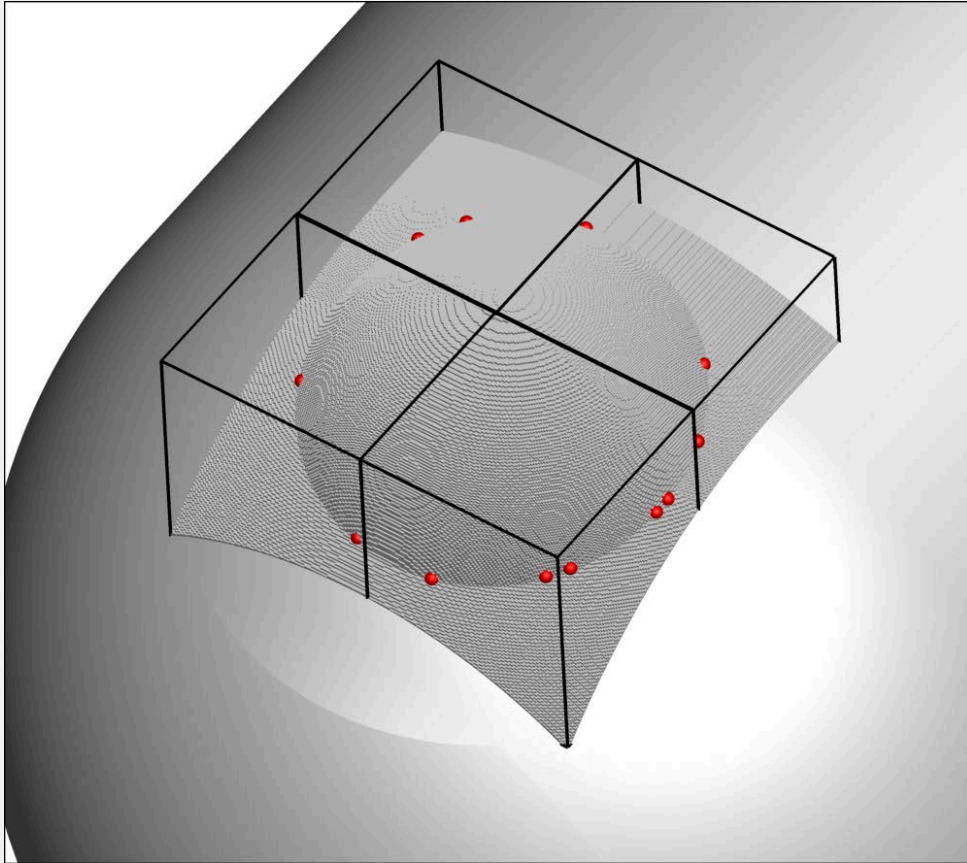
```
$HCUT NAME = 'bump_hcutC',  
      HOFFSET = 1  
      CNX = 512,  
      CNY = 512,  
      CNZ = 512,  
      CARTX = 9.6, 11.6,  
      CARTY = -0.9, 1.1,  
      CARTZ = 1.6, 2.9,  
      CORNERO = 00001111,  
$END
```

Set **\$HCUT** input: **HOFFSET=1**

Result: 0 orphans

- **HOFFSET** grows every hole cut by this **HCUT** one cell in each j,k,l direction
- See also **HOFFSTJ**, **HOFFSTK**, **HOFFSTL**

Protuberance: Use AHCUT and NAHCUT = 4



```
$AHCUT NAME = 'bump_hcutD',
    NAHCUT = 4,
    CNX = 128,
    CNY = 128,
    CNZ = 128,
    CARTX = 9.6, 11.6,
    CARTY = -0.9, 1.1,
    CARTZ = 1.6, 2.9,
    CORNERO = 00001111,
$END
```

Use **AHCUT** to create four course **HCUTs**

Result: 203 orphans

Additional Refinements:

- Use NAHCUT=4, 256x256x256: 104 orphans
- Use NAHCUT=4, 512x512x512: 26 orphans
- Use NAHCUT=4, 128x128x128, HOFFSET=1: 0 orphans
- Use NAHCUT=4, 256x256x256, HOFFSET=1: 0 orphans

`hcut_plot -fringe`

Manual Hole Cutting

Manual hole-cutting from Pegasus4 retained in Pegasus5

- **\$BOUNDARY** and **\$SURFACE** namelists
 - specify a group of zonal surfaces which will cut holes in the specified zones
- **\$BOUNDARY** and **\$BOX** namelists
 - specify a range of x,y,z coordinates of a box
 - box will cut holes in the specified zones
- **\$REGION** and **\$VOLUME** namelists
 - specify a range of j,k,l zonal indices to create a hole
 - specify a range of j,k,l zonal indices to unblank points

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

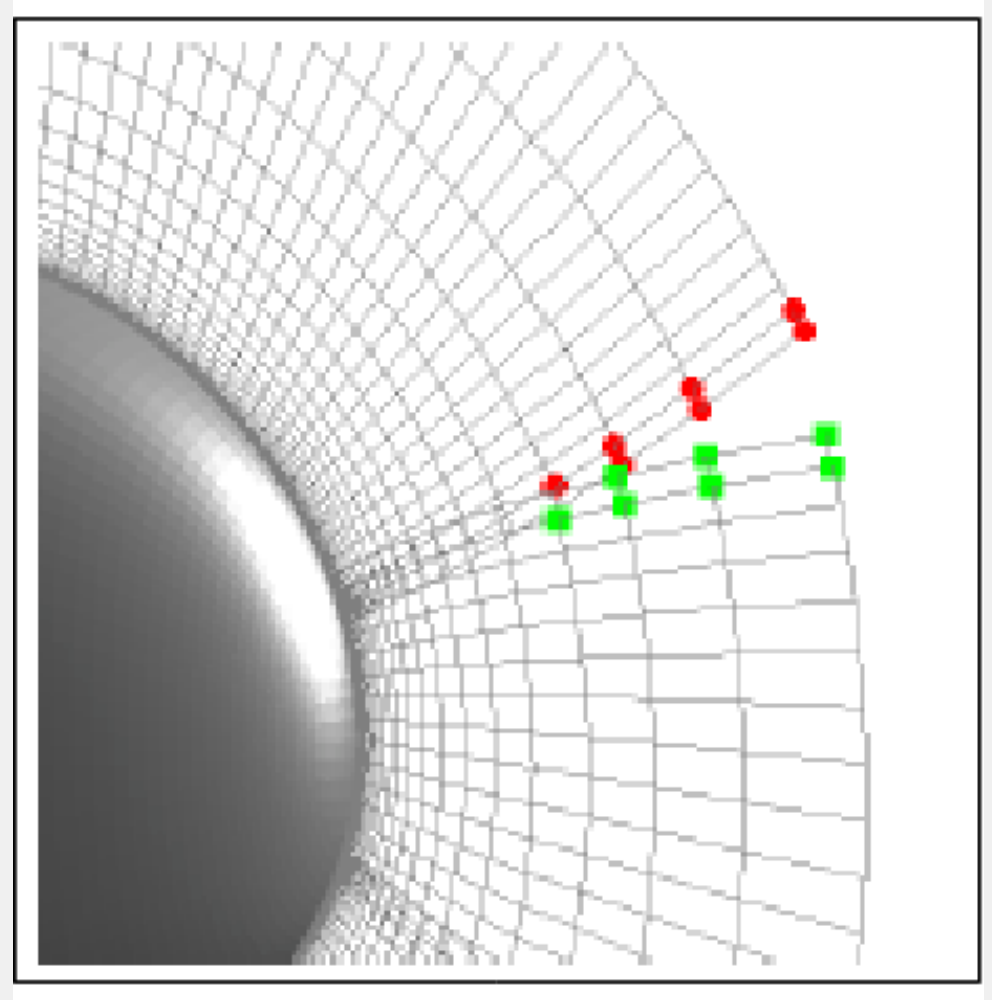
- Orphans are reported in the log file, in the output of `peg_plot`, and using the `peg_orph` program
- Use the `peg_plot` program to create a grid file with iblanks
 - orphan points have `iblack=101`
- Plot orphans using several programs:
 - `plot3d`: Plot using function 3
 - `overgrid`: Grid Diagnostics 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



Utility Codes

- `peg_setup`: creates input file and individual grid files in `X_DIR`
- `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 array
- `peg_plot_center`: creates composite plot3d grid file with iblank array 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
- `dcintegrity`: Runs tests and verifies integrity of the dci file

New Utility Codes Available in Version 5.2

- `hcut_info`: Prints data about each HCUT hole cutter
- `hcut_plot`: Creates plotting files used to examine HCUT hole cutters
 - Run `hcut_plot -help` to see multiple options
- `peg5_plotcpu.pl` Creates plot of CPU usage for each processor for a `pegasus5mpi` run

1. *Introduction and Background*
2. *Overset-grid workflow*
3. *Pegasus5 features and automation*
4. *Overview of usage*
5. **Conclusion**

Conclusion

- **Pegasus5** provides many tools for the oversetting process
- Not a *black-box* procedure: care must be taken to examine the resulting grid system
- See additional documentation in **Pegasus5** package
- Tuesday talk: hybrid **Pegasus5** + DCF
- NASA catalog: <https://software.nasa.gov/software/ARC-15117-1A>

Acknowledgement

- Resources supporting this work were provided by the NASA High-End Computing (HEC) Program through the NASA Advanced Supercomputing (NAS) Division at Ames Research Center

