Improvements to the Pegasus5 Overset CFD Software

Stuart E. Rogers

Applied Modeling and Simulation Branch/Code TNA
NASA Advanced Supercomputing Division
NASA Ames Research Center, Moffett Field, CA

11th Symposium on Overset Composite Grids
October 18th, 2012

Outline

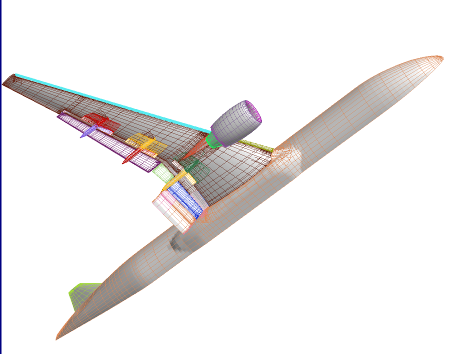
- Introduction
- Support for cell-centered flow solvers
  - Implementation
  - Verification
- Triple-layers of fringe points
- Automatic decomposition into multiple hole-cutters
- Efficiency enhancements for manual hole cuts
- Conclusion

- Add support for cell-centered grids
  - Request from NASA MPCV Orion project
  - DPLR currently supports multi-block and overset grids
  - Leverage grid-generation work done for Overflow analysis
  - Coupled with the Chimera Grid Tools software, provides a very powerful complex-geometry capability
- Three-fringe layers to support higher-order differencing schemes in Overflow
- Complex geometries drive need for improved automation and efficiency
  - Reduce user input for hole-cutting
  - Efficiency improvements in hole-cutting

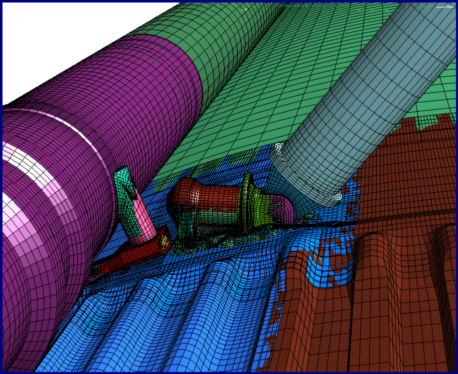
Background: Pegasus5 Usage

Version 5 History: 1998 to present

- Enabled AST Program level-1 milestone: High-Lift Aircraft CFD in 50 days



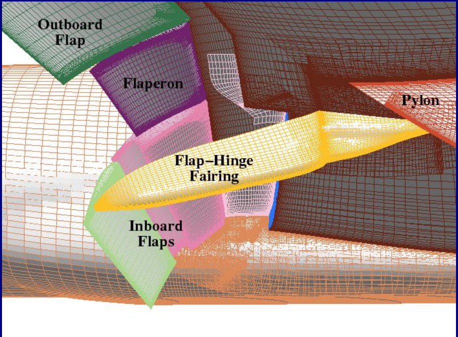
- Enabled AST Program level-1 milestone: High-Lift Aircraft CFD in 50 days
- Space Shuttle Program Return-To-Flight



Background: Pegasus5 Usage

Version 5 History: 1998 to present

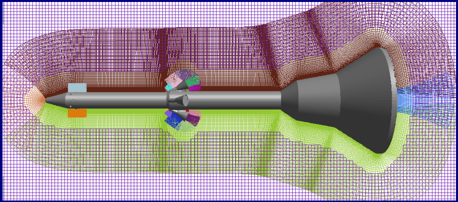
- Enabled AST Program level-1 milestone: High-Lift Aircraft CFD in 50 days
- Space Shuttle Program Return-To-Flight
- Boeing high-lift and cruise CFD analysis



Background: Pegasus5 Usage

Version 5 History: 1998 to present

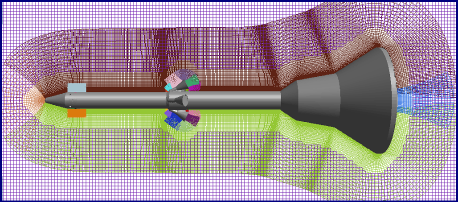
- Enabled AST Program level-1 milestone: High-Lift Aircraft CFD in 50 days
- Space Shuttle Program Return-To-Flight
- Boeing high-lift and cruise CFD analysis
- Orion Launch Abort Vehicle



Background: Pegasus5 Usage

Version 5 History: 1998 to present

- Enabled AST Program level-1 milestone: High-Lift Aircraft CFD in 50 days
- Space Shuttle Program Return-To-Flight
- Boeing high-lift and cruise CFD analysis
- Orion Launch Abort Vehicle
- ... and many more

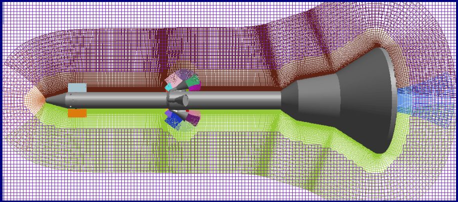


Background: Pegasus5 Usage

Version 5 History: 1998 to present

- Enabled AST Program level-1 milestone: High-Lift Aircraft CFD in 50 days
- Space Shuttle Program Return-To-Flight
- Boeing high-lift and cruise CFD analysis
- Orion Launch Abort Vehicle
- ... and many more

- Distributed to over 350 outside organizations and users



Background: Pegasus5 Features and Capabilities

Current Version 5.1

- Automatic hole-cutting
  - Multi-step hybrid method using indirect and direct hole cutting
  - Cartesian hole maps provide indirect representation of hole shape
  - Line-of-sight test using surface-grid elements: direct refined hole cutting
- Hole optimization through use of "level 2" interpolation
- Internal projections between overlapping surface grids
- Finds best interpolation stencil through exhaustive search
- Parallel execution using MPI on shared and distributed memory systems
- Automatic restart capability
- Maintains manual hole-cutting capability from Pegsus4

- Computationally expensive, this is mitigated by parallelization
- Stand-alone program: cannot be used for time-accurate moving-body problems
- Overflow cannot run in DCF mode and use the Pegasus5 XINTOUT file
    - Cannot use automatic off-body Cartesian grids
    - Cannot use Overflow adaptive grid refinement

Cell-Centered Grids

Support for DPLR CFD code

DPLR Code

- Data Parallel Line Relaxation
- Navier-Stokes hypersonic flow solver
- Structured, 3D, cell-centered, finite volume
- DPLR uses the dirtLib overset library from Ralph Noack
- Overset stencils and iblanks read in "dci" format

Overset Requirements

- Input-grid coordinates located at vertex locations
- Hole points defined at the cell-centers
- Fringe points and donor points defined at cell-centers
- Work with hybrid multi-block/overset meshes: donor cell-centers span multi-block boundaries

Cell-Centered Implementation

General Approach

- Generate cell-centered meshes with ghost cells
- Identify multi-block connectivity
- Projection: interpolate from wall vertices to cell centers
- ADT: build trees using cell-centers
- Interpolation: search using cell centers
- Hole cutting:
    - All hole-cutting operations performed on vertex nodes
    - Transfer blanking to cell-centers
    - Cell-center is blanked if any 8 surrounding vertices blanked
- Identify fringe cell-centers: hole fringes, outer-boundary fringes, level-2 fringes
- Output: all stencils and iblank array written to ".dci" file

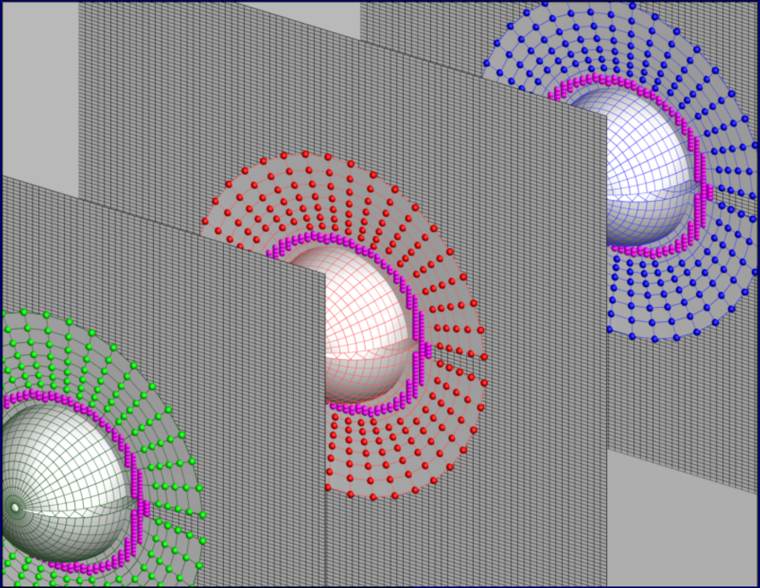
Verification with *dcintegrity* Program

Cell-center version of XINtegrity

- Reads the pegasus5.dci file and the cell-center coordinates
- Verifies all fringe indices are valid
- Verifies all donor indices are valid
- Verifies all donor weights add to 1.0 for each fringe point
- Verifies all hole points are surrounded by fringe points
- Verifies that all fringe points are marked in the iblank array
- Verifies the interpolation stencils
  - Error = Interpolated donor coordinates - recipient coordinates

Three Spheres Test Case

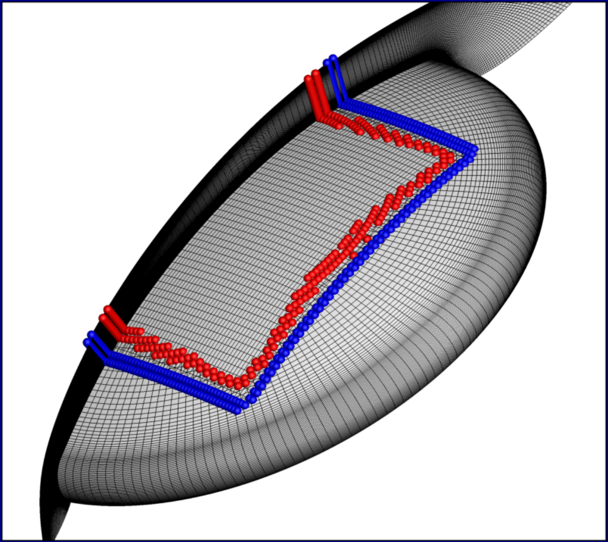
Periodic-grid verification test



Orion Heatshield Test Case

Courtesy of Chun Tang/NASA Ames/TSA

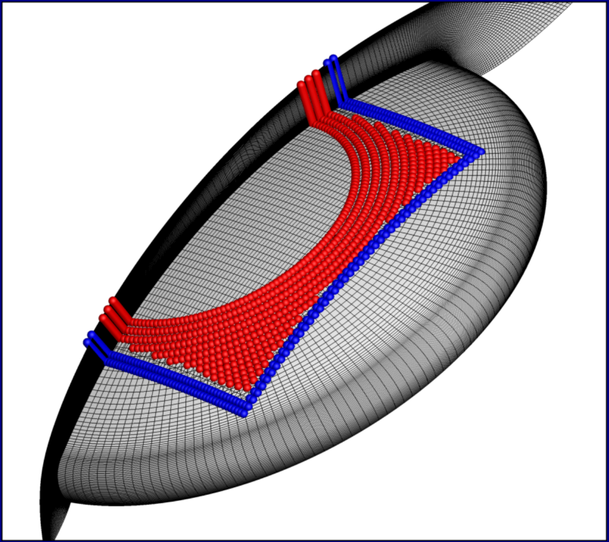
- 2 Zones
- 1.3M points
- Suggar Fringes



Orion Heatshield Test Case

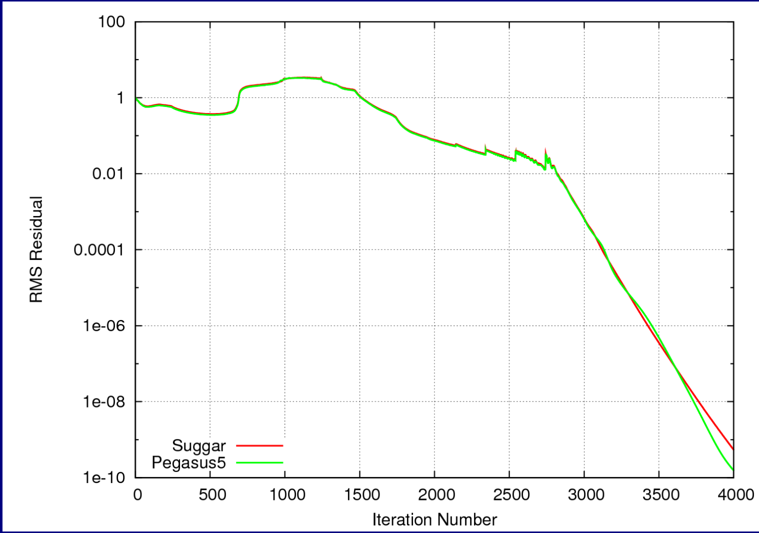
Courtesy of Chun Tang/NASA Ames/TSA

- 2 Zones
- 1.3M points
- Pegasus5 Fringes



Orion Heatshield Test Case

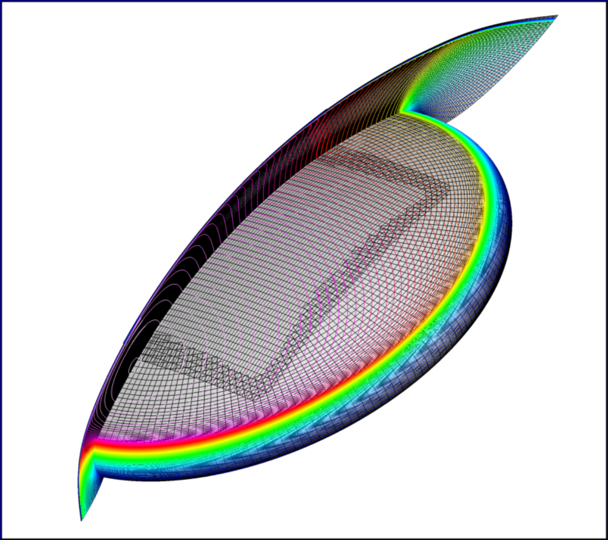
DPLR Convergence: Mach = 23.5



Orion Heatshield Test Case

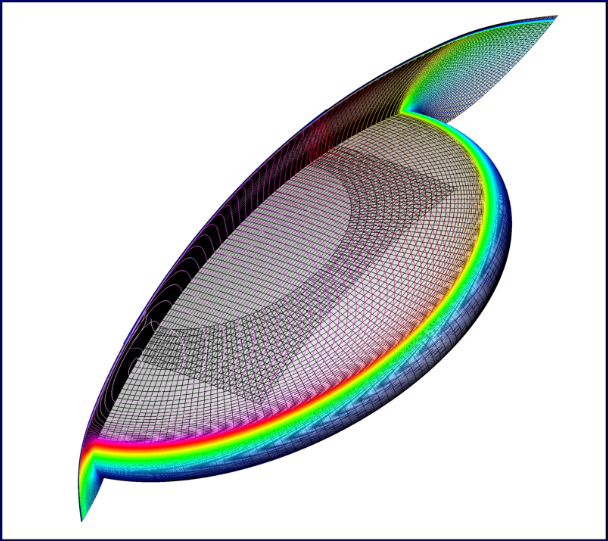
DPLR Results: Mach = 23.5

- Suggar Pressure



DPLR Results: Mach = 23.5

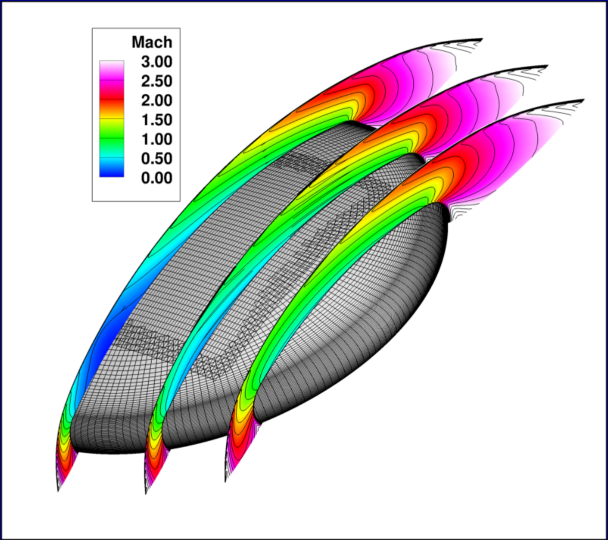
- Pegasus5 Pressure



Orion Heatshield Test Case

DPLR Results: Mach = 23.5

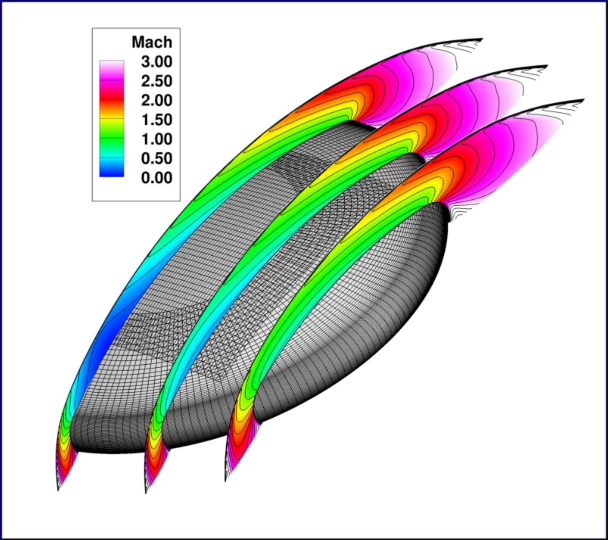
- Suggar Mach



Orion Heatshield Test Case

DPLR Results: Mach = 23.5

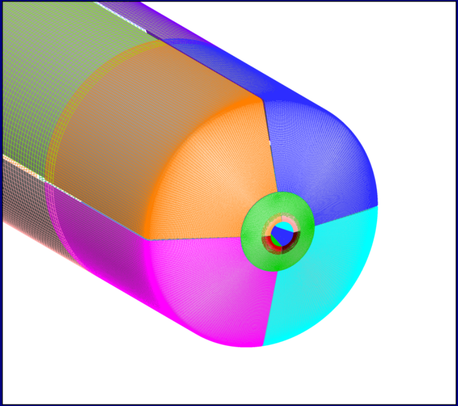
- Pegasus5 Mach



Supersonic Retro-Propulsion (SRP) Test Case

Courtesy of Kerry Zarchi/NASA Ames/TSA

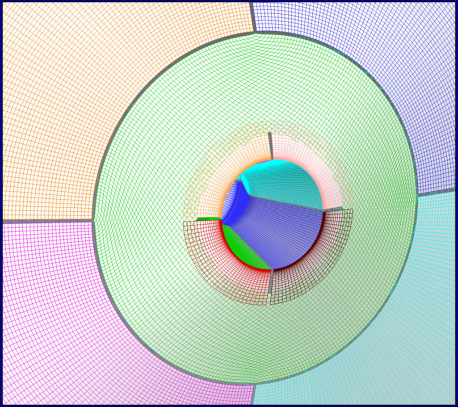
- 29 zones
- 20.8M points
- Multi-block and Overset
- Pegasus5 wallclock = 100 sec
- 12 Intel Xeon CPUs
- One manual hole cut



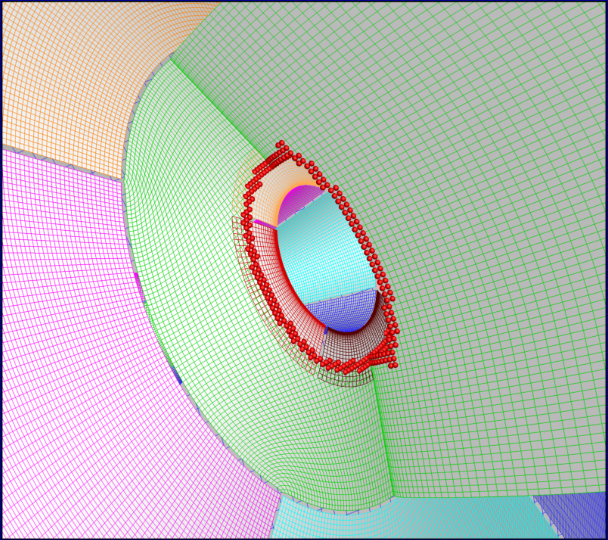
Supersonic Retro-Propulsion (SRP) Test Case

Courtesy of Kerry Zarchi/NASA Ames/TSA

- 29 zones
- 20.8M points
- Multi-block and Overset
- Pegasus5 wallclock = 100 sec
- 12 Intel Xeon CPUs
- One manual hole cut



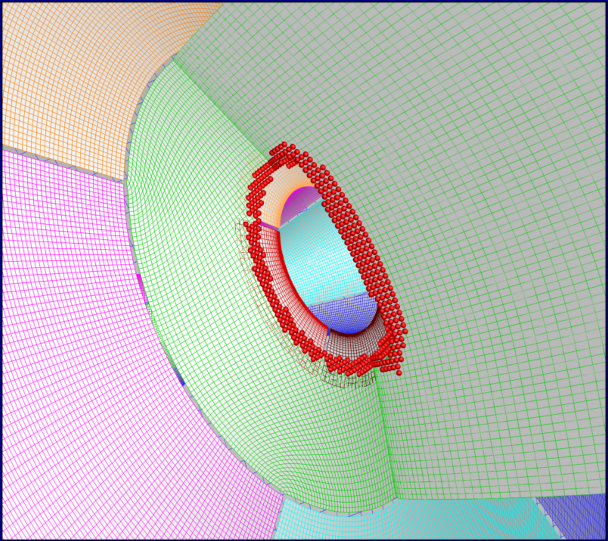
- Suggar Fringes



SRP Test Case

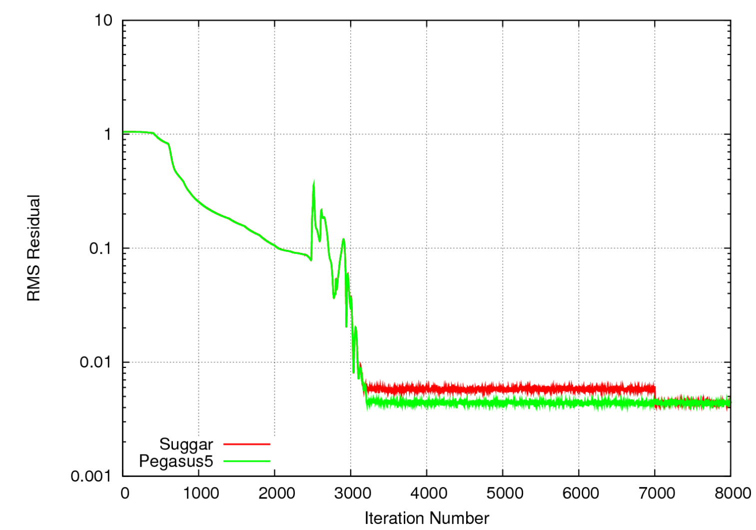
Overset Fringes

- Pegasus5 Fringes



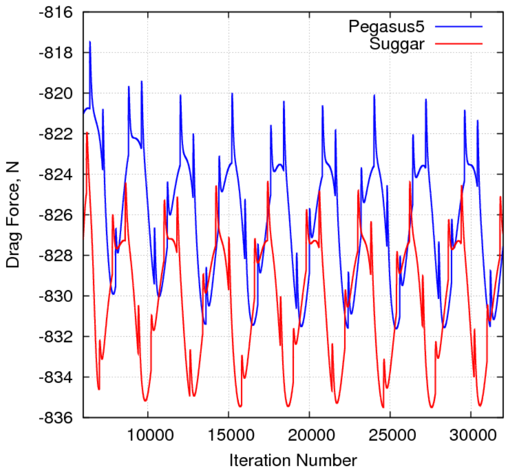
SRP Test Case

DPLR Convergence: Mach=2.4



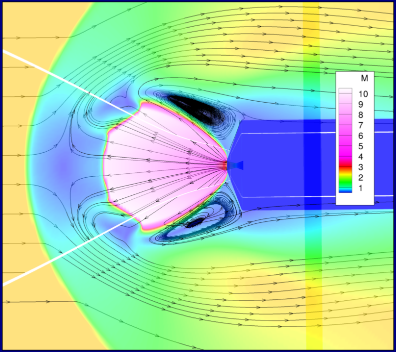
SRP Test Case

DPLR Drag Force Convergence: Mach=2.4

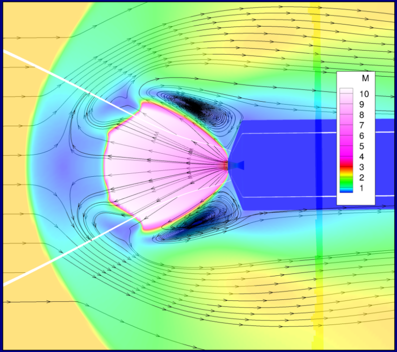


SRP Unsteady Flow Fields

Pegasus5

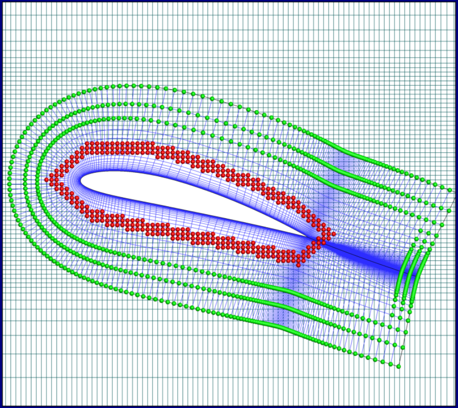


Suggar



Triple-Layers of Fringe Points

- Can request three layers for hole fringes and/or outer-boundary fringes
- Reports numbers of orphans in each of the first, second, and third layers
- Second and third layer orphans can be turned back into interior points

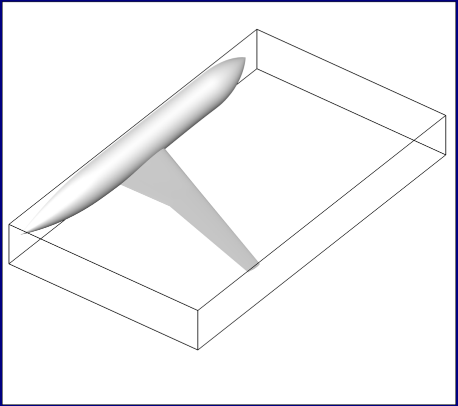


Automatic HCUT Creation

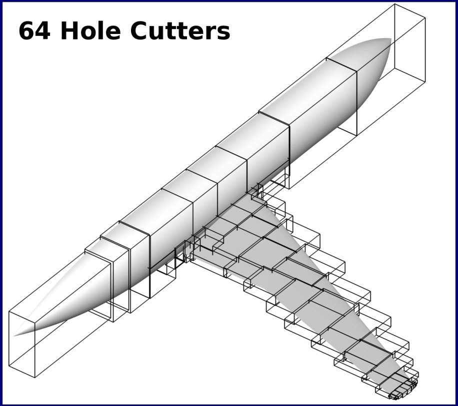
- Enhance auto hole cutting using domain decomposition
- Current default: one hole-cutter
  - Automatically creates bounding-box around all solid-walls
  - Cut holes in all zones
- Current recommended practice is to create multiple HCUT hole-cutters: requires manual specification of bounding boxes

Goal: Automatic Decomposition To Fit The Geometry

One Hole-Cutter

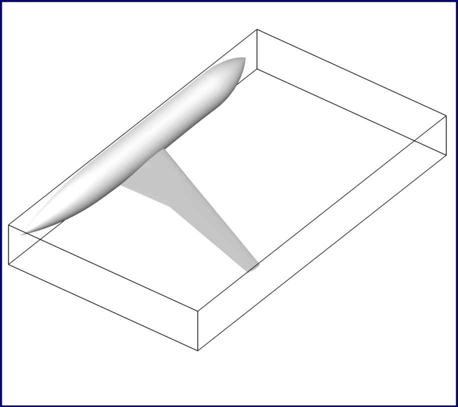


64 Hole-Cutters

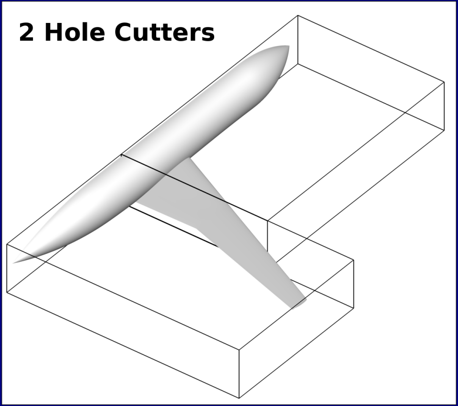


Approach

- Recursively split the domain

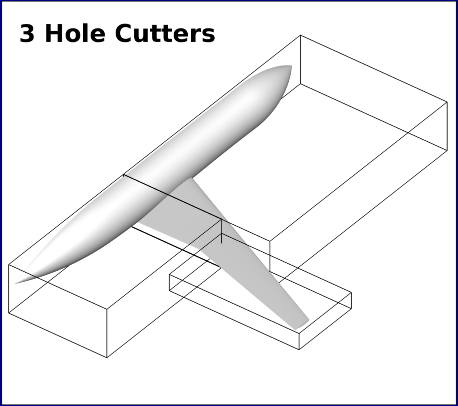


- Recursively split the domain
  - Split the box in the longest dimension



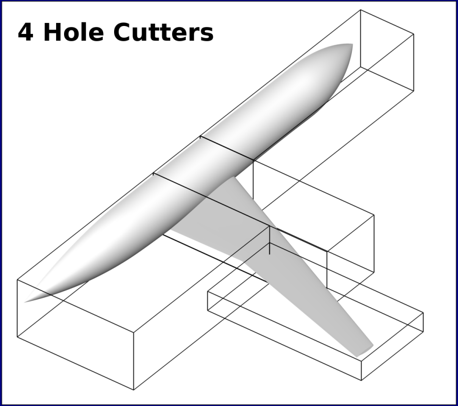
Approach

- Recursively split the domain
  - Split the box in the longest dimension
  - Split the box with the most surface-grid points



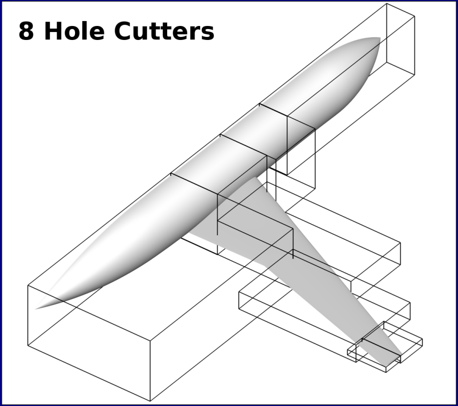
Approach

- Recursively split the domain
  - Split the box in the longest dimension
  - Split the box with the most surface-grid points



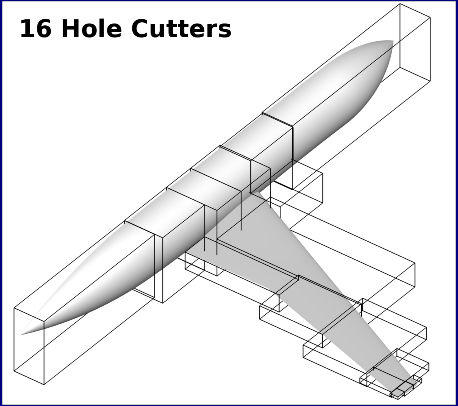
Approach

- Recursively split the domain
  - Split the box in the longest dimension
  - Split the box with the most surface-grid points



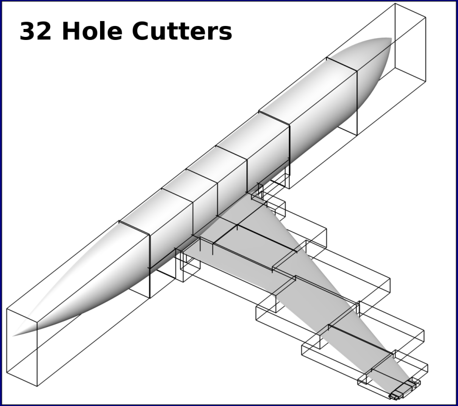
Approach

- Recursively split the domain
  - Split the box in the longest dimension
  - Split the box with the most surface-grid points



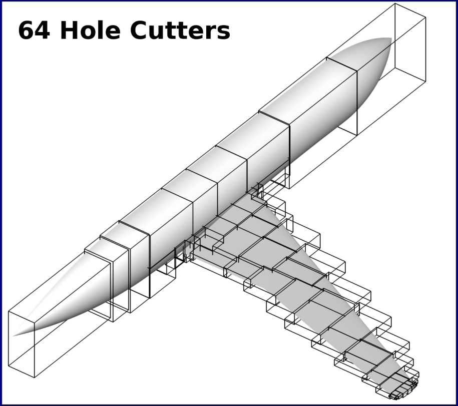
Approach

- Recursively split the domain
  - Split the box in the longest dimension
  - Split the box with the most surface-grid points
  - Never create a box completely inside



Approach

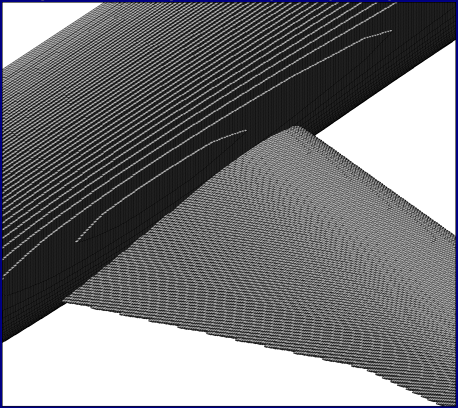
- Recursively split the domain
  - Split the box in the longest dimension
  - Split the box with the most surface-grid points
  - Never create a box completely inside



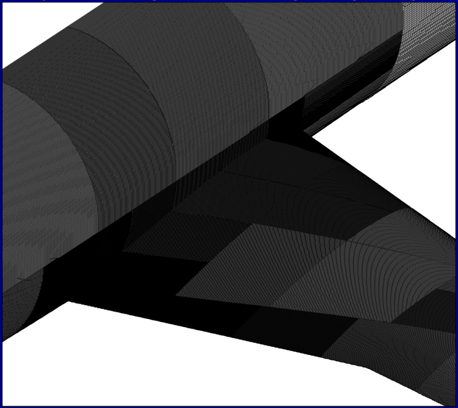
Wing-Body Test Case: Cartesian Fringe Elements

Ratio of Total Cartesian Volume = 10.1

One Hole-Cutter: 512x512x512



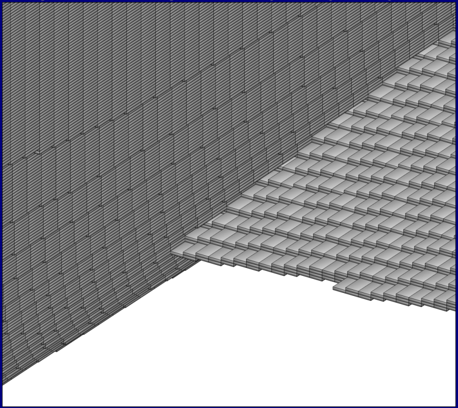
64 Hole-Cutters: 128x128x128



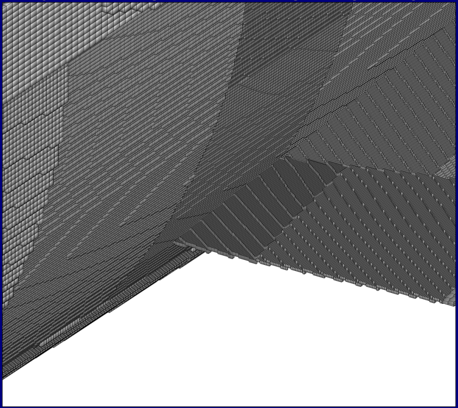
Wing-Body Test Case: Cartesian Fringe Elements

Ratio of Total Cartesian Volume = 10.1

One Hole-Cutter: 512x512x512

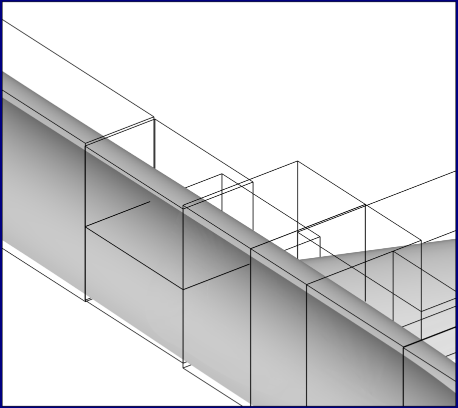


64 Hole-Cutters: 128x128x128



Modifications to Painting Algorithm

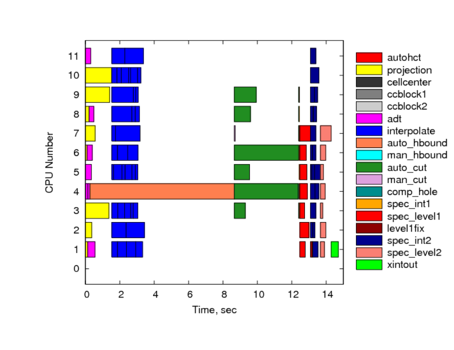
- Required improvements to painting algorithm
- Detect which hole-cutter box corners are inside, which are outside
- Newly created corners use line-of-sight test to determine inside or outside
- Seed the painting only on the outside corners



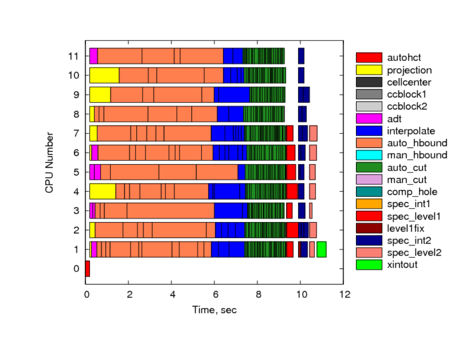
Wing-Body Test Case: Parallel CPU Usage

12 Intel Xeon CPUs

One Hole-Cutter: 512x512x512

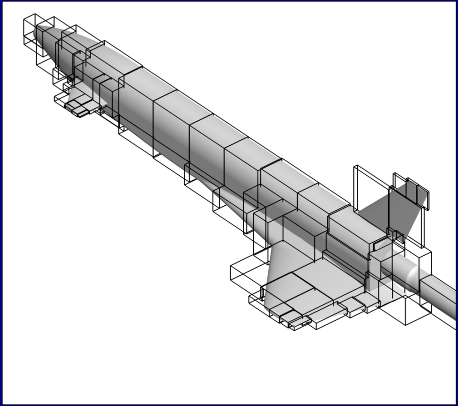


64 Hole-Cutters: 128x128x128

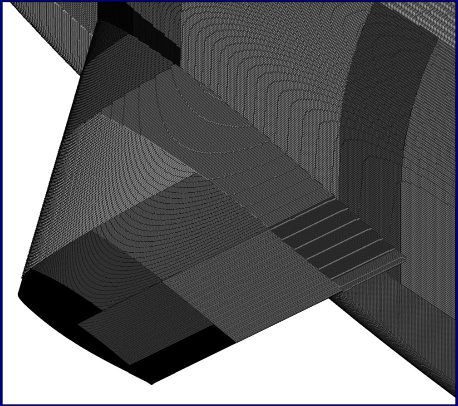


Liquid Glide-Back Booster Example

64 Hole-Cutters



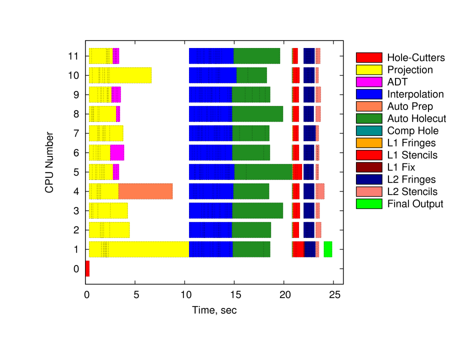
Fringe Elements



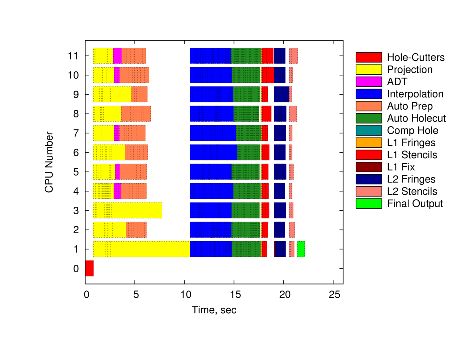
Liquid Glide-Back Booster: Parallel CPU Usage

12 Intel Xeon CPUs

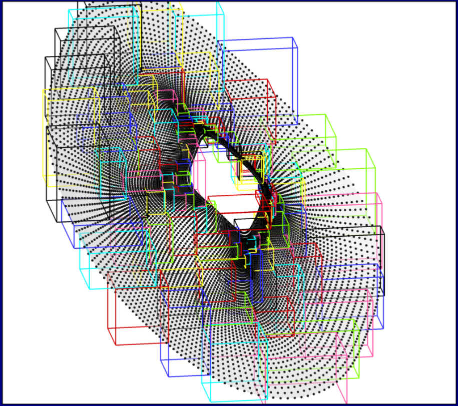
One Hole-Cutter: 512x512x512



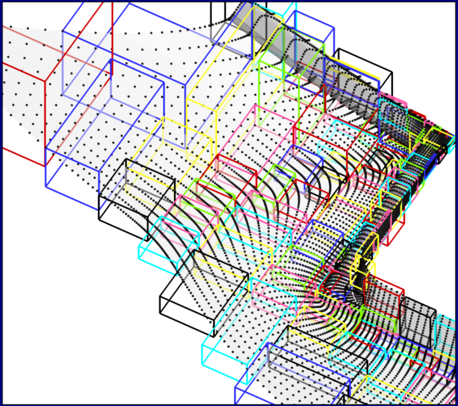
64 Hole-Cutters: 128x128x128



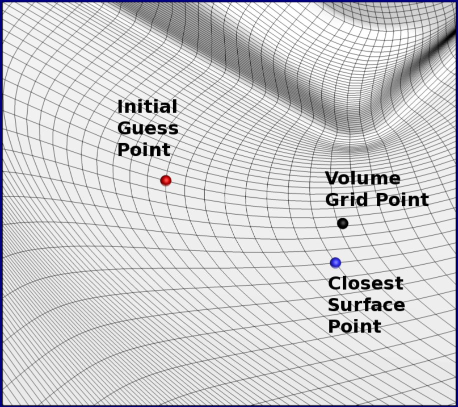
- Taken from the Walldist program, now part of Overflow
- Wigton, Parlette, Biedron, Rumsey, Jespersen
- Exact Search
- Used to replace the old exhaustive search algorithm
- Speed-up: 10-20 times faster



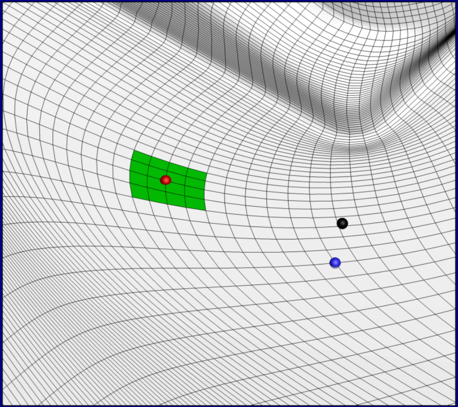
- Construct boxes recursively: cut along longest axis, equal number of points in each half
- Search algorithm:
  - Compute distance to each bounding box
  - Find closest distance for each point in closest box
  - If next-closest box distance < closest-point distance, search that box
  - Repeat last step until box-distance > point-distance



- Provide initial guess for closest surface point



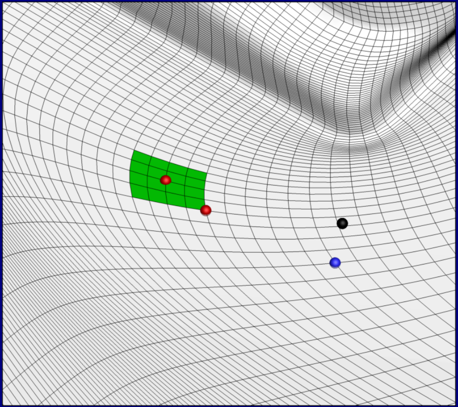
- Provide initial guess for closest surface point



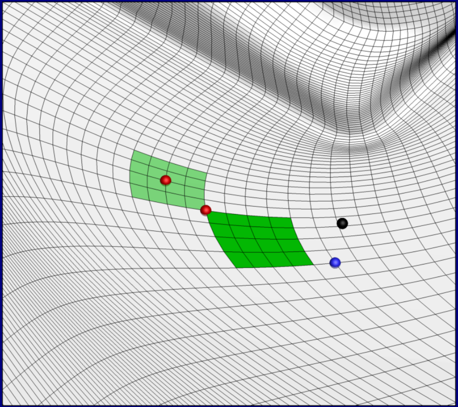
Manual-Hole Cut Efficiency Improvements

Marching Patch Algorithm

- Provide initial guess for closest surface point
- Compute distance for surface points in subset patch



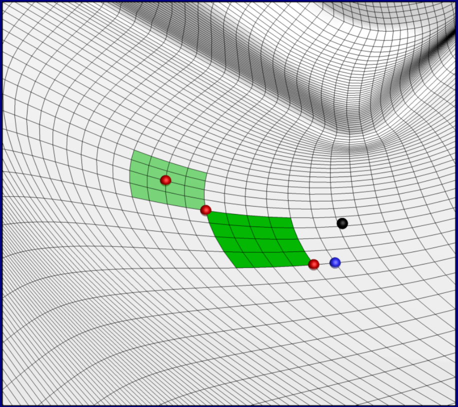
- Provide initial guess for closest surface point
- Compute distance for surface points in subset patch
- Iteratively march the surface patch



Manual-Hole Cut Efficiency Improvements

Marching Patch Algorithm

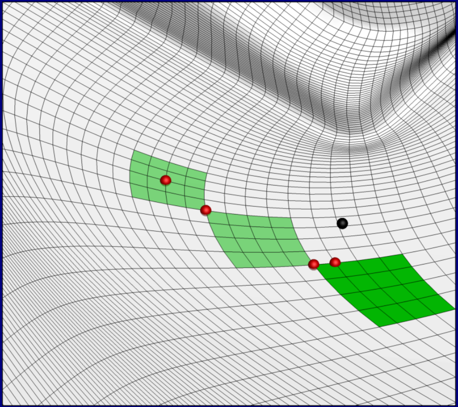
- Provide initial guess for closest surface point
- Compute distance for surface points in subset patch
- Iteratively march the surface patch



Manual-Hole Cut Efficiency Improvements

Marching Patch Algorithm

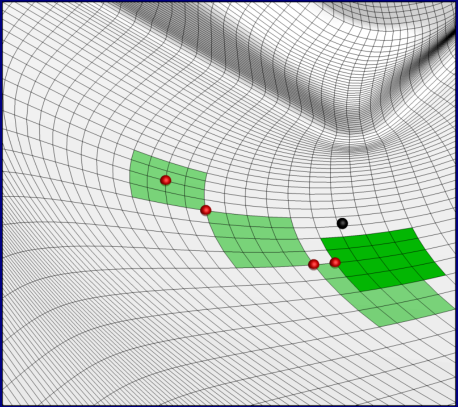
- Provide initial guess for closest surface point
- Compute distance for surface points in subset patch
- Iteratively march the surface patch



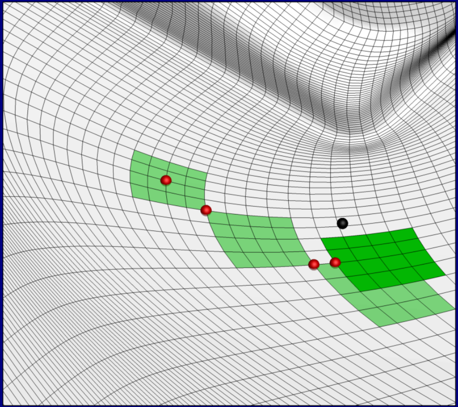
Manual-Hole Cut Efficiency Improvements

Marching Patch Algorithm

- Provide initial guess for closest surface point
- Compute distance for surface points in subset patch
- Iteratively march the surface patch
- Stop when minimum-distance point does not change



- Provide initial guess for closest surface point
- Compute distance for surface points in subset patch
- Iteratively march the surface patch
- Stop when minimum-distance point does not change
- Inexact near surface creases, highly curved surfaces
- Provided as an option in the code
- Speed-up: 100-200 times faster



Conclusion

- Cell-centered grids and DPLR support
  - Produces holes and interpolation stencils for cell-centers
  - Verified operation and results using several test cases
- Implemented a domain-decomposition approach to create HCUT hole-cutters:
  - More efficient use of Cartesian elements
  - Improved parallel efficiency
- Version 5.2 of Pegasus will soon be available for β-testing