Advanced Modeling and Simulation Seminar Series
NASA Ames Research Center, May 25, 2017



Progress in Strand/Cartesian Overset CFD simulations using CREATE™-AV Helios



Distribution Statement A, Approved for public release; distribution unlimited.
Review completed by the AMRDEC Public Affairs Office (PR3051, 09 May 2017)

TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

Jay Sitaraman — Parallel Geometric Algorithms, LLC

Beatrice Roget — Science and Technology Corporation

Vinod Lakshminarayan — Science and Technology Corporation

Andy Wissink

Aviation and Missile Research, Development and Engineering Center
Moffett Field, CA

TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

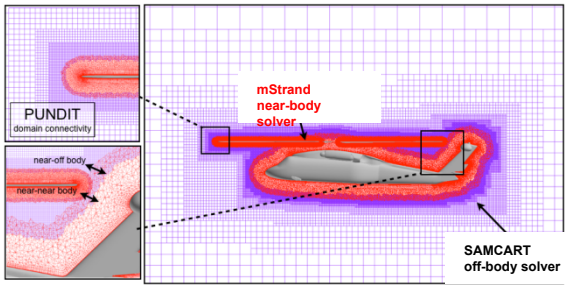
Outline

- Introduction
- Motivation for this work
- Methodology
  - Strand mesh generation
  - Domain connectivity for colliding strands
- Results
  - Single-bladed rotor
  - DLR-F6 from DPW-III
  - UH60 Fuselage
  - Common Research Model
  - UH-60 fuselage + rotor
- Concluding observations

Helios Code

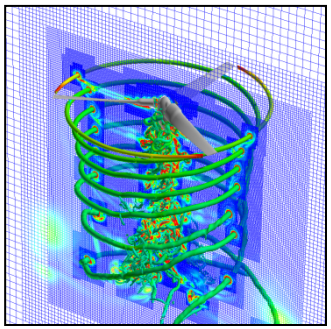
HPCMP CREATE™ AV Helios Helicopter Overset Simulations

Dual Mesh Paradigm



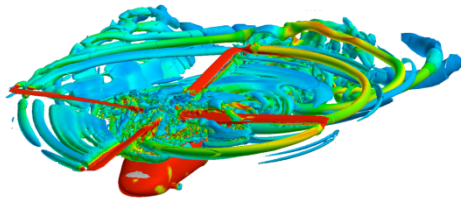
Near-body – mStrand, Overflow, FUN3D, …
Cartesian off-body - SAMCart

Adaptive Mesh Refinement



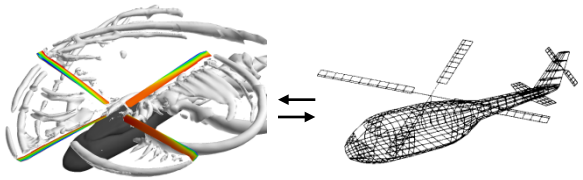
To resolve wake

Moving Body Overset



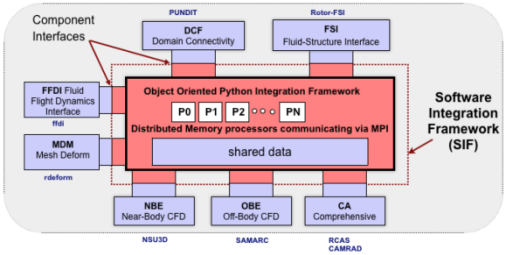
Rotor-Fuselage and Multi-rotor moving mesh support

CFD/CSD Coupling



RCAS and CAMRAD Structural Dynamics and Trim coupling

Advanced Software Infrastructure



Python-based infrastructure readily supports addition of new software

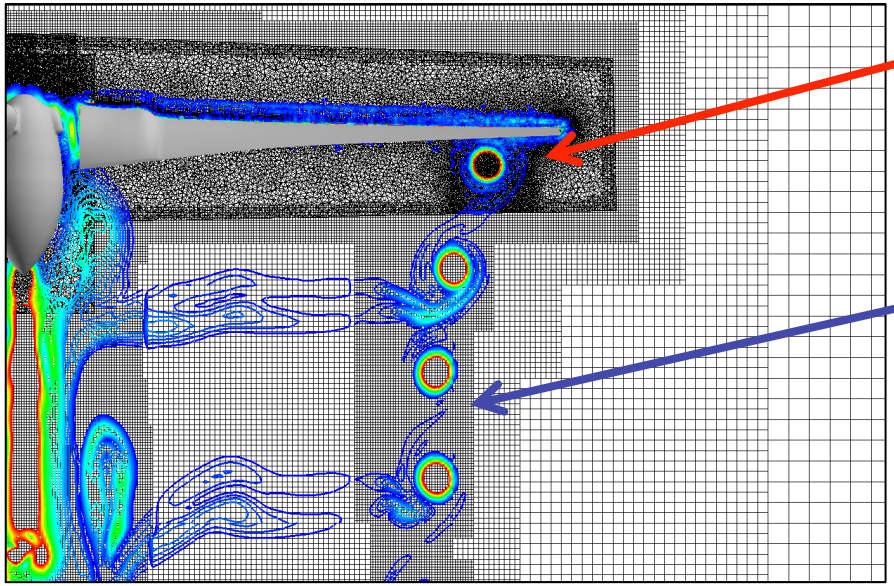
High Performance Computing



Runs on HPC hardware with focus on parallel scalability

- High-fidelity rotorcraft simulation for government and industry
- Introduces strand grid framework in v7 released in 2016

Motivation for Dual-Mesh

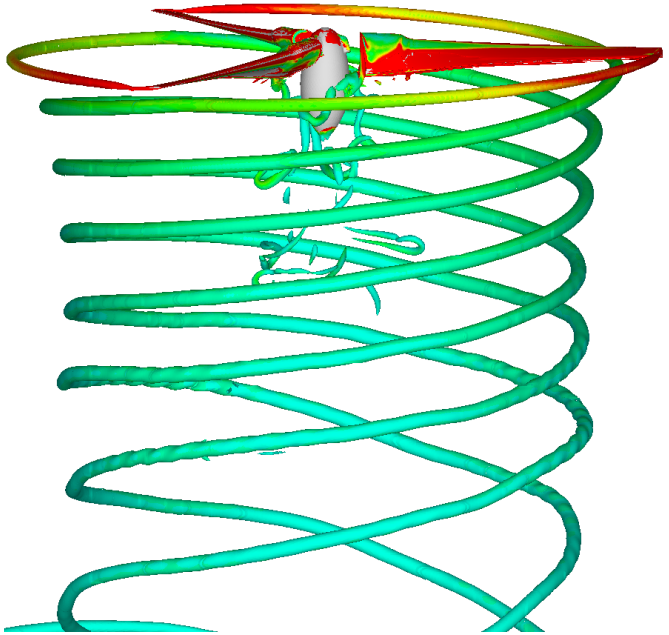


Body-conforming "near-body"

- Resolve near-wall viscous flow
- Complex geometries

Cartesian "off-body"

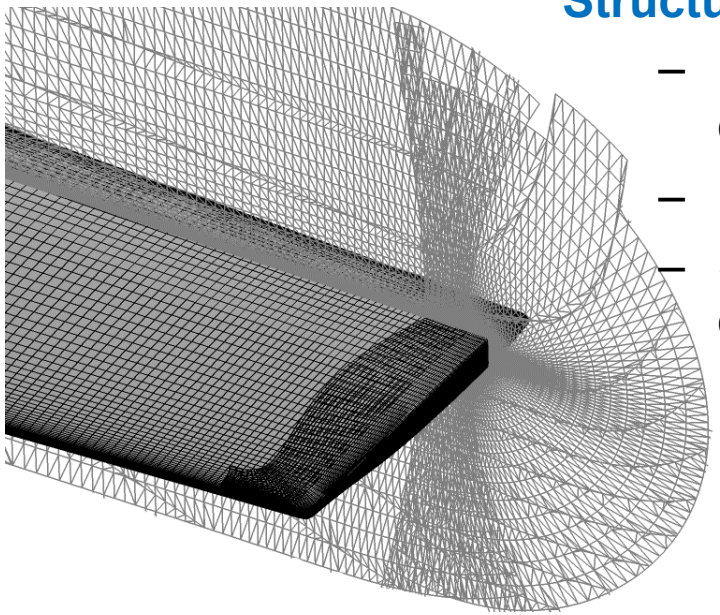
- Computationally efficient
- High order accuracy
- Adaptive mesh refinement



→ **Preserves wake to enable accurate predictions:**

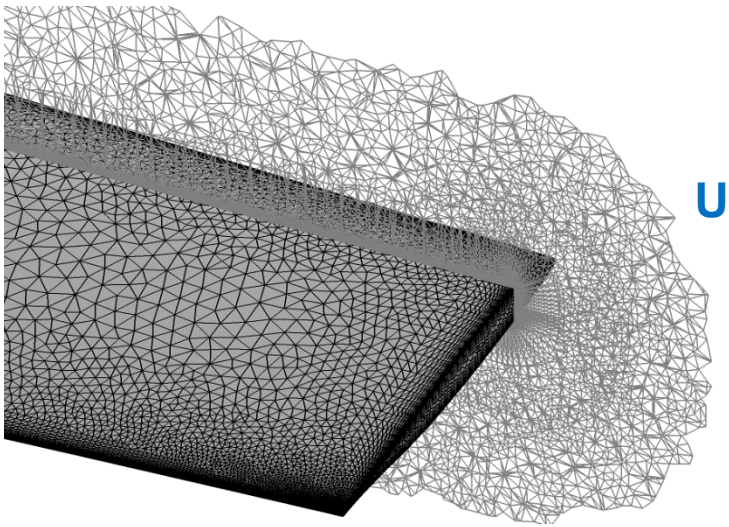
- Interactional aero
- Noise
- Ground effect

Near-Body Mesh Types



Structured mesh:

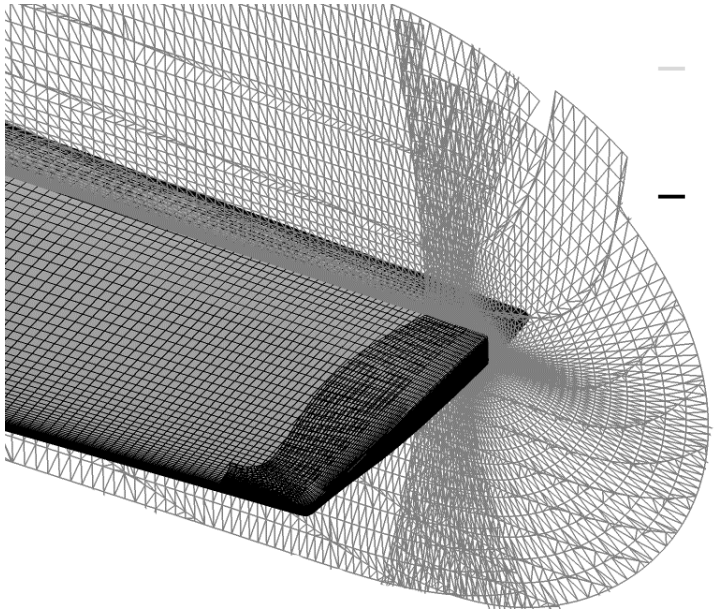
- Mesh generation can be tedious for complex geometries
- High-order flow solution possible
- Scalability problems for domain connectivity on very large-scale

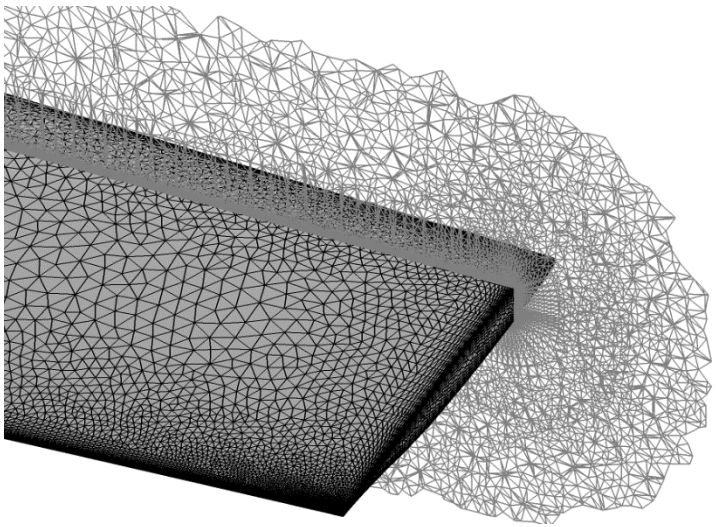


Unstructured mesh:

- More automated mesh generation
- Generally slower than structured grid counterparts
- Scalability problems for domain connectivity on very large-scale

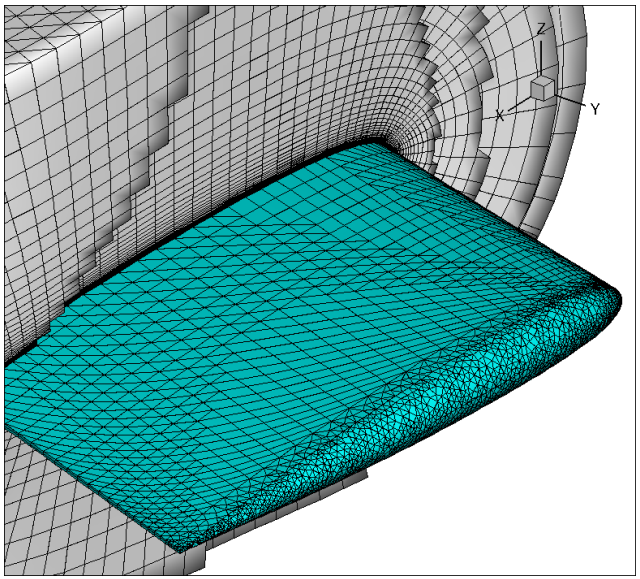
Near-Body Mesh Types





- Mesh generation can be tedious for complex geometries
- **High-order flow solution possible**

Strand mesh:
Semi-structured
Scalable domain connectivity (compact storage)



- **More automated mesh generation**
- Not easily amenable to high-order flow solution
- Poor scalability for domain connectivity on very large-scale

Strand/Cartesian Overset Meshes

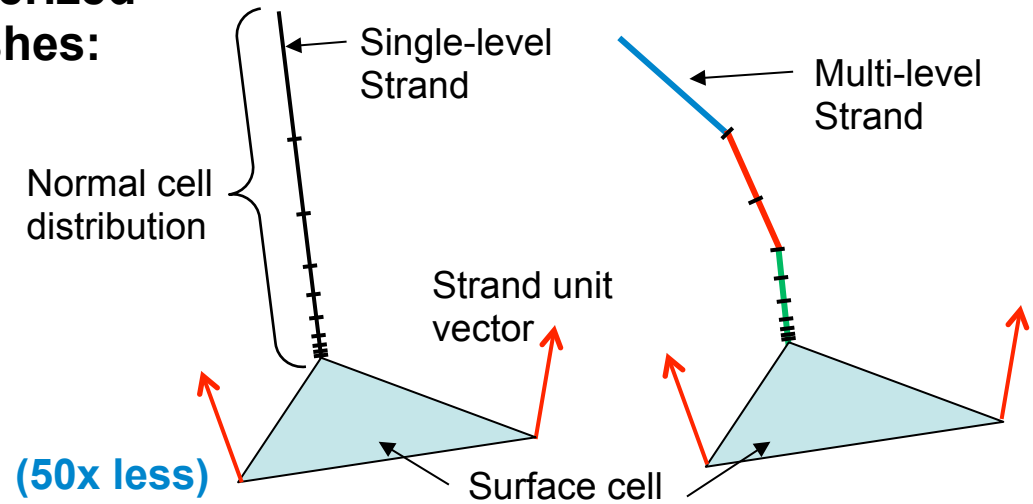
- **Strand meshes allow parameterized representation of volume meshes:**
  - **surface tessellation**
  - **pointing vectors**
  - **normal distribution**

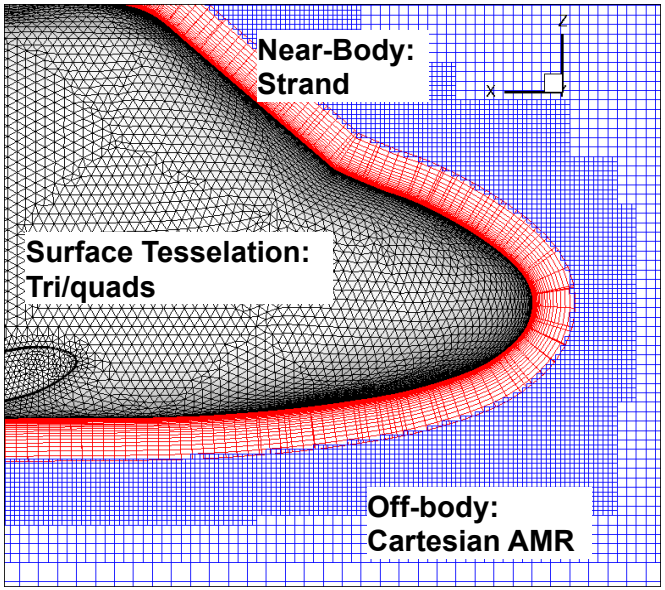
Example: 50 million node grid:

- **Unstructured → 4.2 GB**
- **Single-level Strand → 80 MB (50x less)**
- **Multi-level with CRS → 100 MB (40x less)**

- **Strands provides a pathway to**
  - **Automated scalable volume meshing from CAD/surface tessellation**
  - **Scalable Domain Connectivity (compact mesh in each proc.)**
  - **Efficient line-based solver technology**

Strand Mesh Structure:

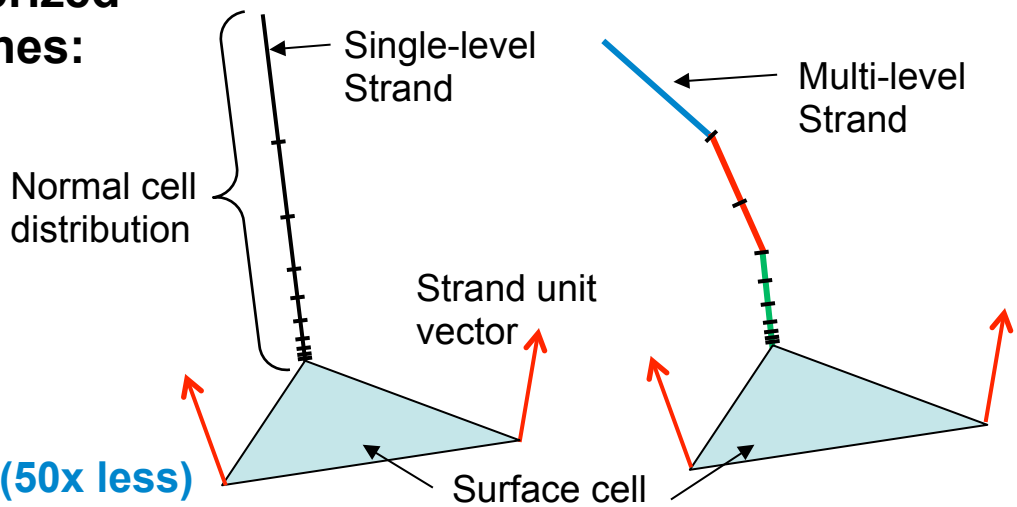




Strand/Cartesian Overset Meshes

- **Strand meshes allow parameterized representation of volume meshes:**
  - **surface tessellation**
  - **pointing vectors**
  - **normal distribution**

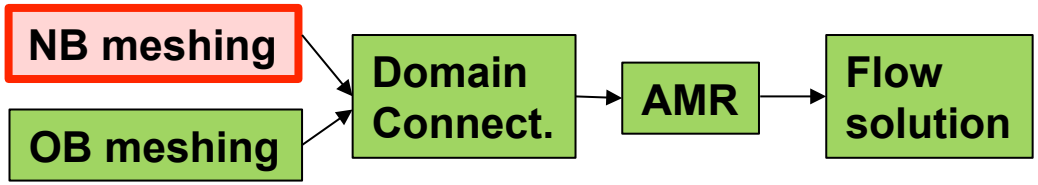
Strand Mesh Structure:



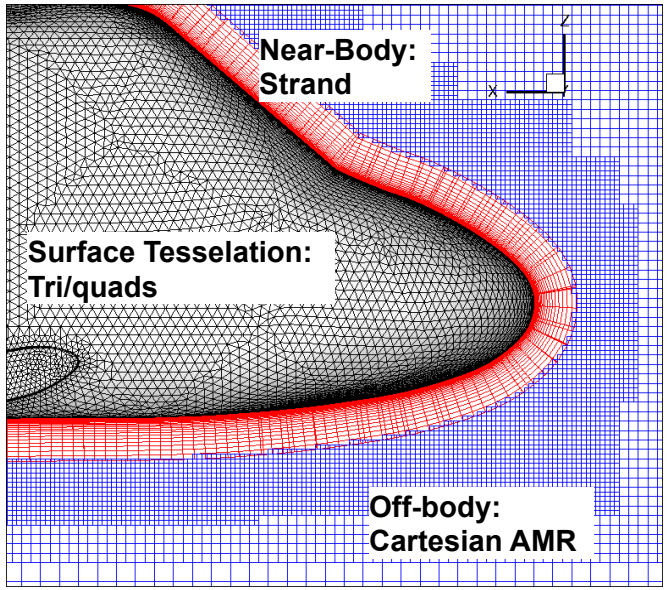
Example: 50 million node grid:

- **Unstructured → 4.2 GB**
- **Single-level Strand → 80 MB (50x less)**
- **Multi-level with CRS → 100 MB (40x less)**

- **Strands provides a pathway to**
  - **Automated scalable volume meshing from CAD/surface tessellation**



Fully automated tasks



Strand Meshing/Solution Challenges

- **Need an efficient flow solver that takes advantage of the strand structure.**
  - ***mStrand (Lakshminarayan et al., AIAA 2015, AIAA 2016, AIAA 2017, Computers & Fluids 2017, AHS 2017)***

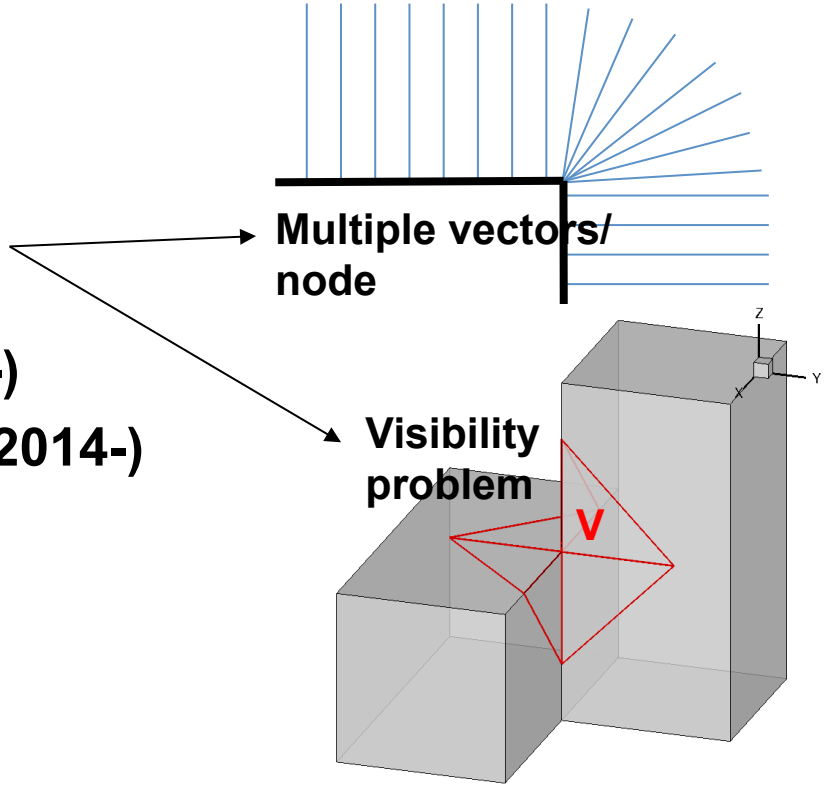
- **Mesh generation**
  - **Have to develop methods to place strands in actual geometry with concave, convex and concave/convex features**

- **Need an efficient domain connectivity for intra-mesh (within a strand mesh), inter mesh (between various meshes) and inter domain (between Strand and Cartesian)**

History of Strand Technology

Meshing and Domain Connectivity

- Proposed by R. Meakin et.al
  - Sprout code by W. Chan (2007)
- Multi-strand mesh generation
  - MOSS code by B. Haimes (2013-)
  - mStrandGen code by B. Roget (2014-)
- Scalable domain connectivity
  - OSCAR by J. Sitaraman (2012-)



Solver technology

- Many research efforts by A. Katz et al.
  - Strand3D, Strand3DFC (2008-2015)
- Production code by V. Lakshminarayan
  - mStrand (2015-), integral part of Helios V7
- Current work, improvements to meshing and domain connectivity, towards achieving automated solutions for real problems.

15+ conference papers
8 journal articles

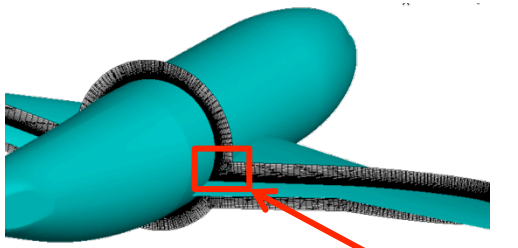
Meshing Method

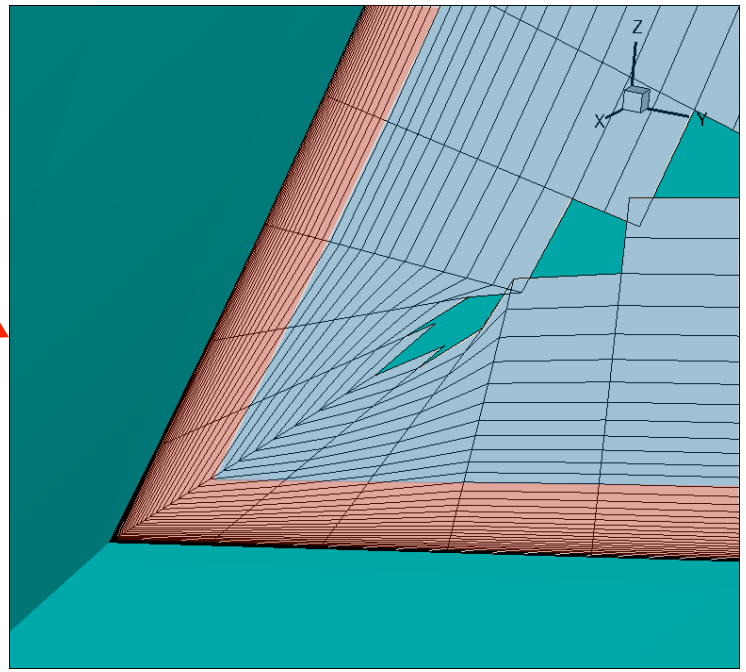
Meshing Method Overview

In order to allow meshing of complex geometries, Strand Meshes are built in 2 steps:

1. **Inner layer, can be very close to surface (covering boundary layer): must be fully closed (no self-intersection)**

2. **Outer layer, extends larger distance to allow interfacing with Cartesian grid: can be self-intersecting**



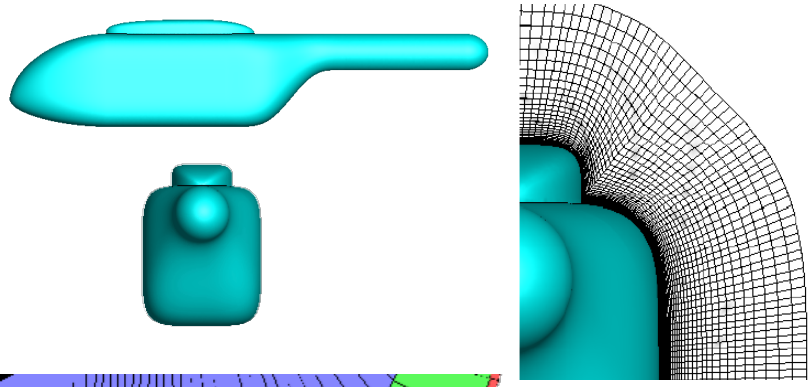


Outer layer:
- Extrusion from 1st layer
- intra-mesh domain connectivity to clear self-intersections

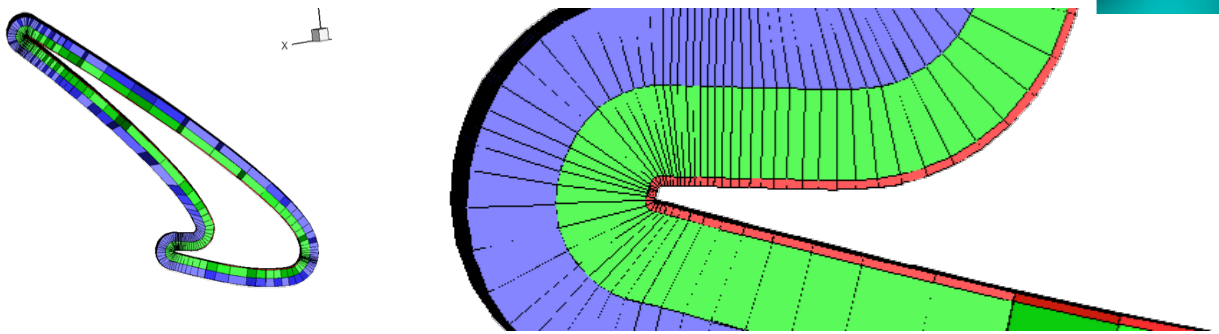
Inner layer:
- fully closed Strand mesh
- may consist of a few levels (< 5)

Inner Layer

- Typically single-level strands for simple geometries:

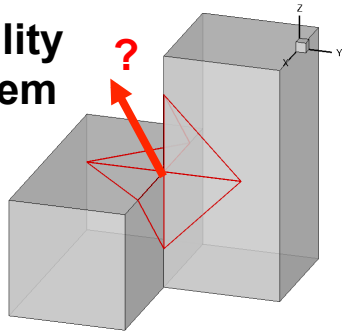


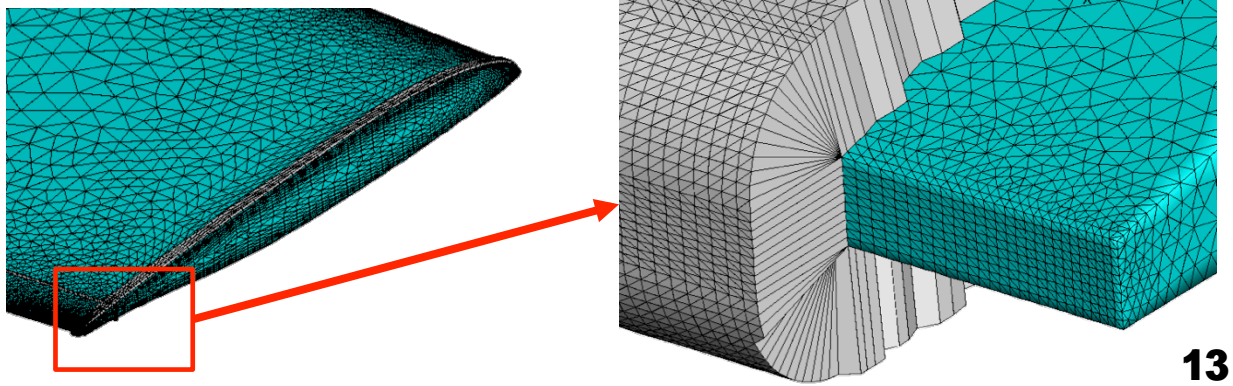
- Can be segmented into several levels for more complex geometries:



- Initial segment can be multi-stranded to solve visibility/coverage issues

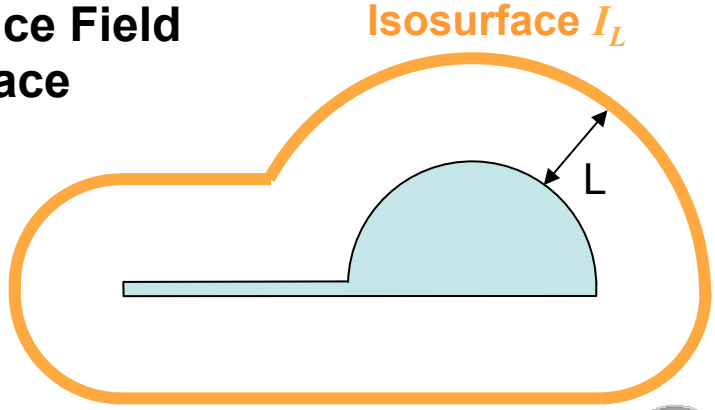
Visibility problem

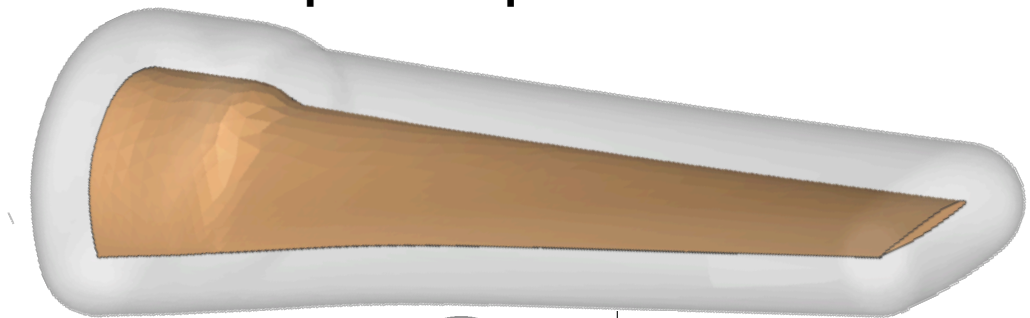


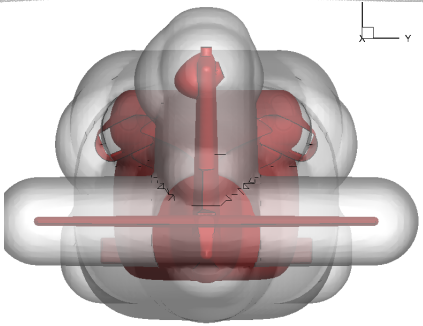


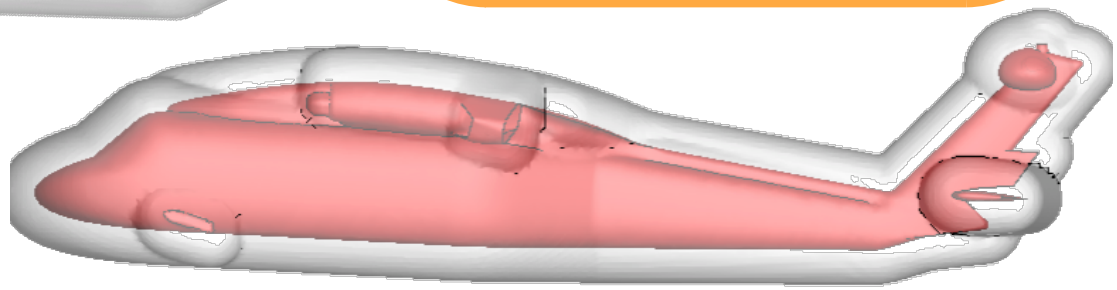
Meshing Method per Level

Based on the Isosurface of Minimum Distance Field
= locus of points equidistant from base surface

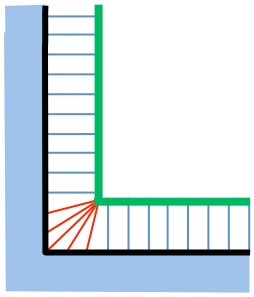


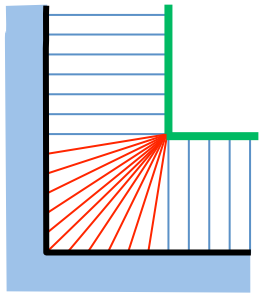






Idea: connect surface node to its closest point on I_L
→ strands in concave areas are automatically bent:



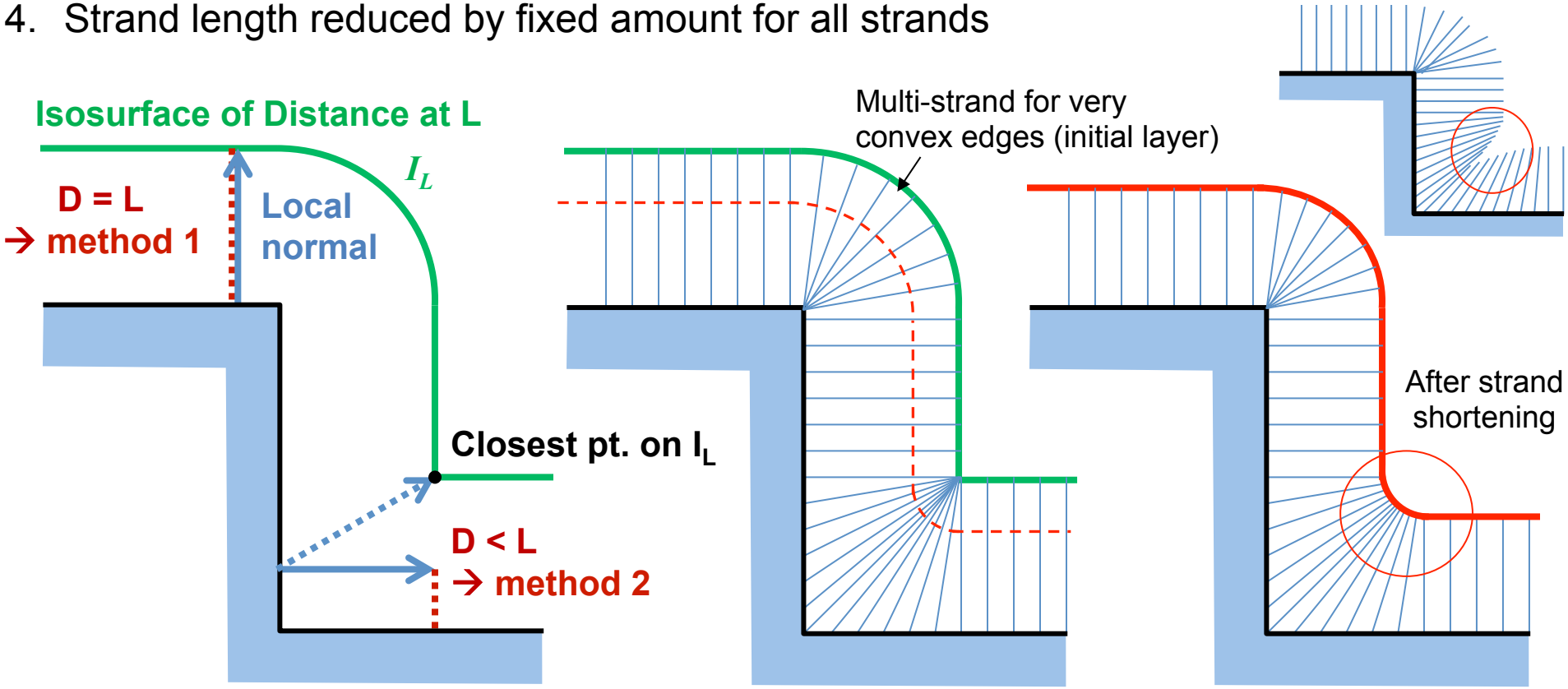


Closest Vertex on the Isosurface of Distance Field (CLOVIS)

Sitaraman, et al. "Progress in Strand Mesh Generation and Domain Connectivity for Dual-Mesh CFD simulations." *55th AIAA Aerospace Sciences Meeting*, 2017

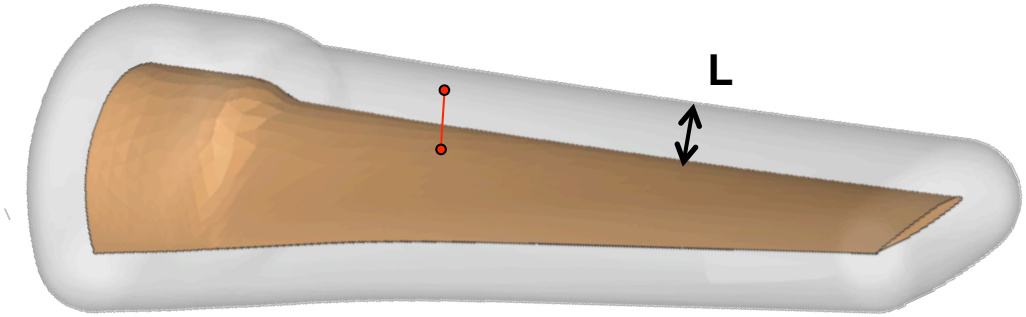
Clovis Method (1)

1. Strands created following local **best visibility** direction, length L **(method 1)**
2. Distance from strand tip to surface computed, D
3. If D<L, strand direction re-computed by finding closest point to iso-surface of distance at L **(method 2)**
4. Strand length reduced by fixed amount for all strands



Clovis Method (2)

CLOsest Vertex on the ISo-surface of Distance Field



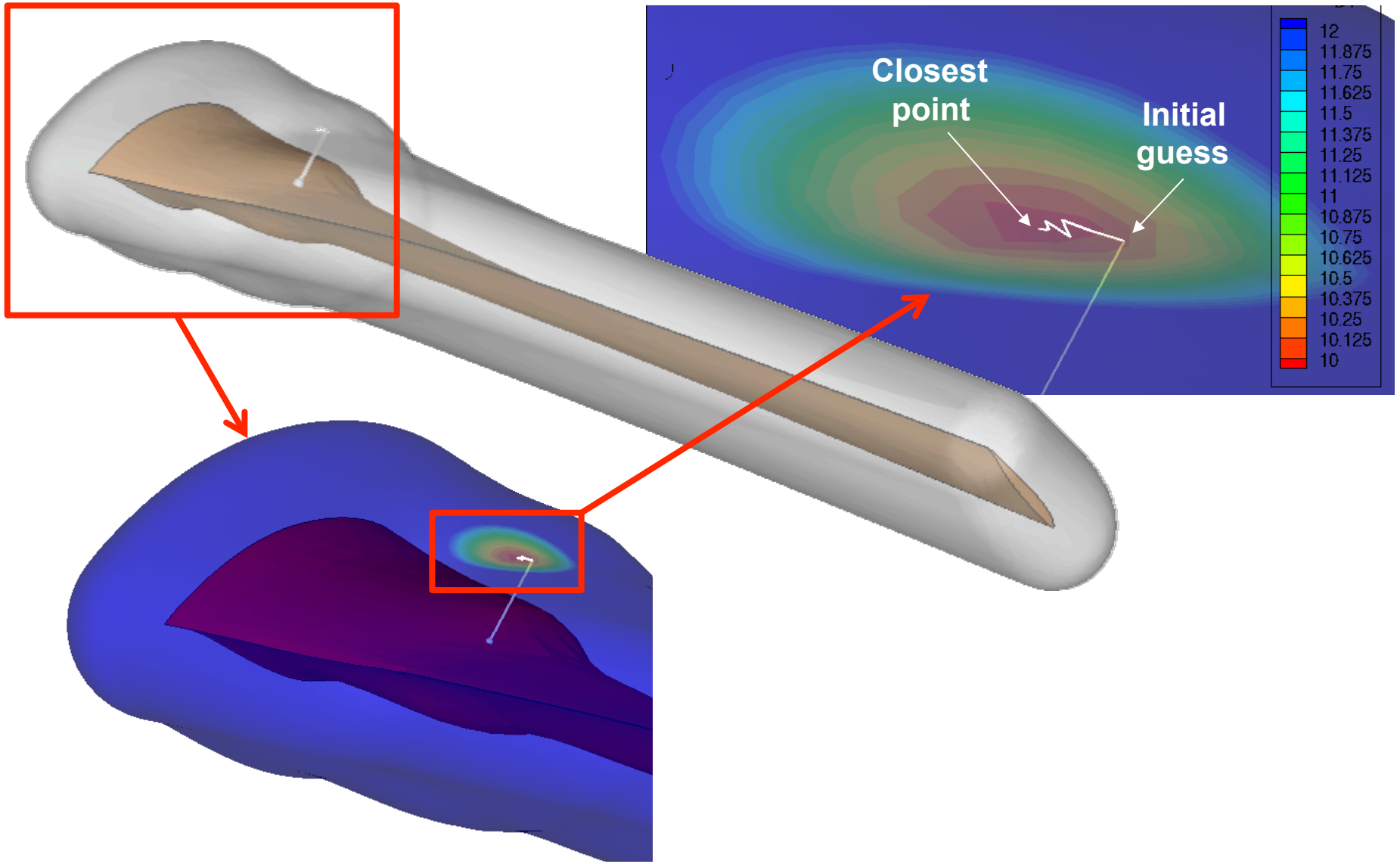
Problem definition

- Minimize strand length, constraining the strand end point to be at a distance of L from the surface. Optimization problem with a strong non-linear constraint.
- Can solve it approximately using a tessellated distance field iso-surface (Wissink 2014). Vertex location is sensitive to the size of iso-surface tessellation. Need an exact solution to be robust.
- Solve the continuous optimization problem, for each surface point x_0

$$\min_x g(x) = L\ (|x - x_0|),\quad g(x) = \min_{y \in \partial\Omega}\ (|x - y|)$$

- Requires a distance field evaluation routine to compute
  $g(x)$ = distance between point x and the surface tessellation.
  Use custom optimization algorithm (in the paper)

Clovis Method (3)



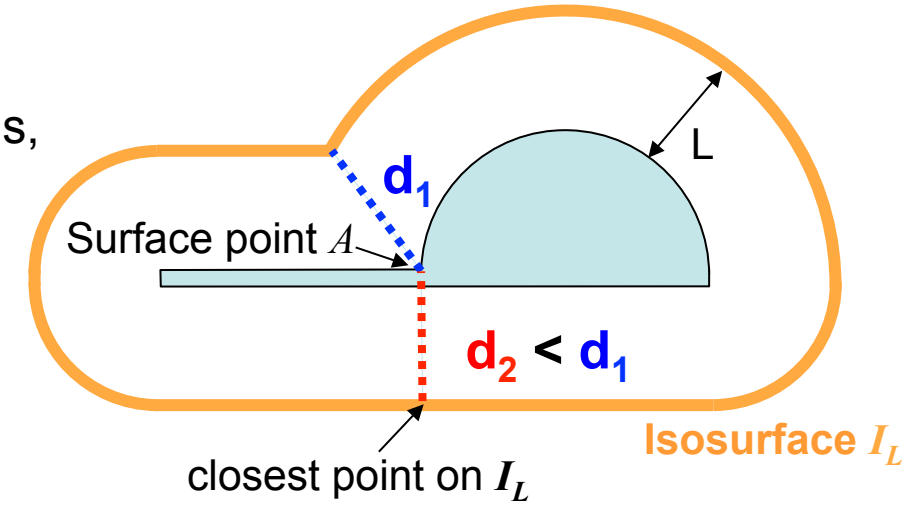
Direction of Best Visibility

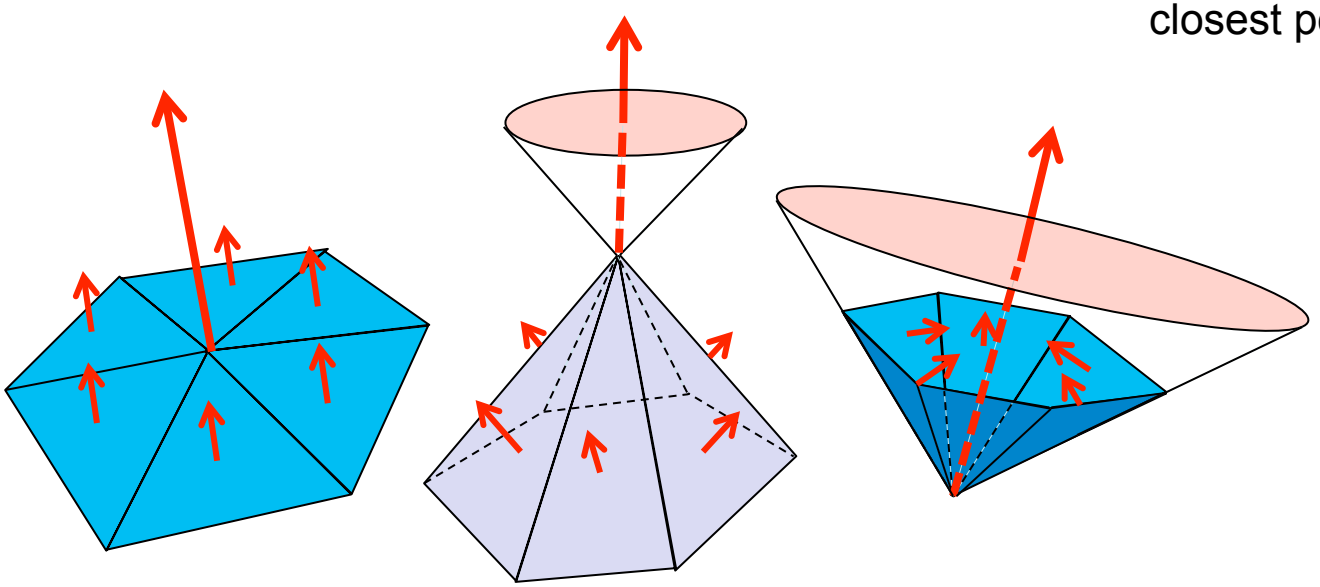
Problem:

- when body thickness has abrupt variations, CLOVIS can yield invalid mesh (surface intersection)

Solution:

- Use Direction of Best Visibility (minimizes max. angle with neighbor face normals)





Flat surface: Full visibility

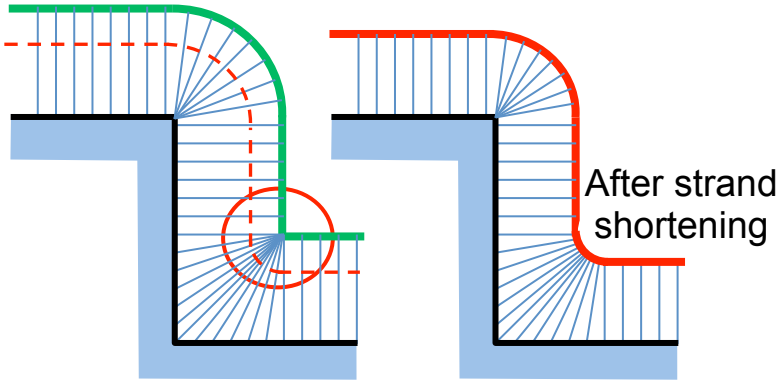
Convex node **Concave node**
Partial visibility

If CLOVIS solution is outside visibility cone: Use best visibility direction instead

Elastic Smoothing on the Isosurface of Distance

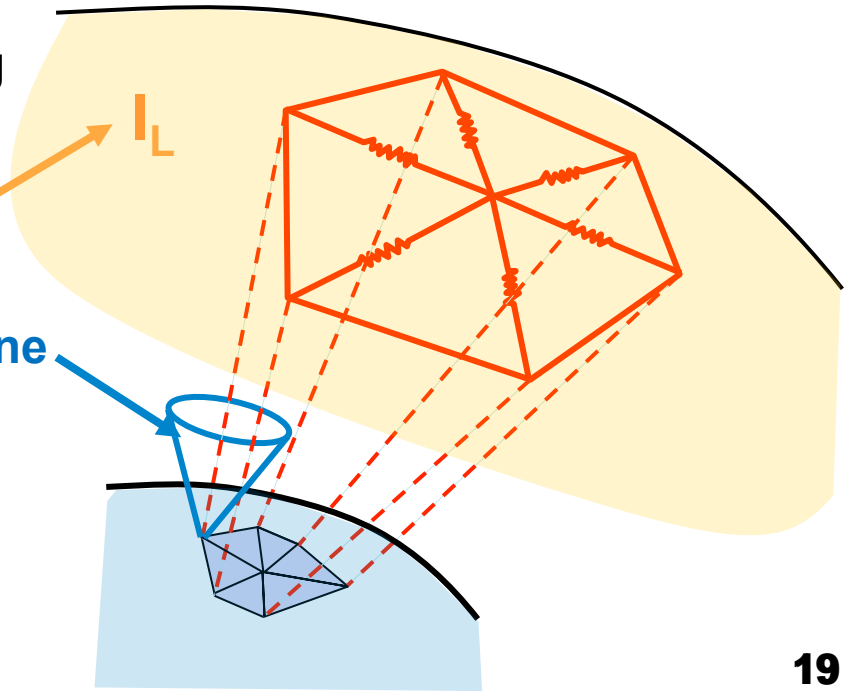
Initial Strand mesh obtained from CLOVIS / best visibility direction:

- can have invalid volumes
- often presents too much point clustering on the envelope surface (poor stretch ratio)



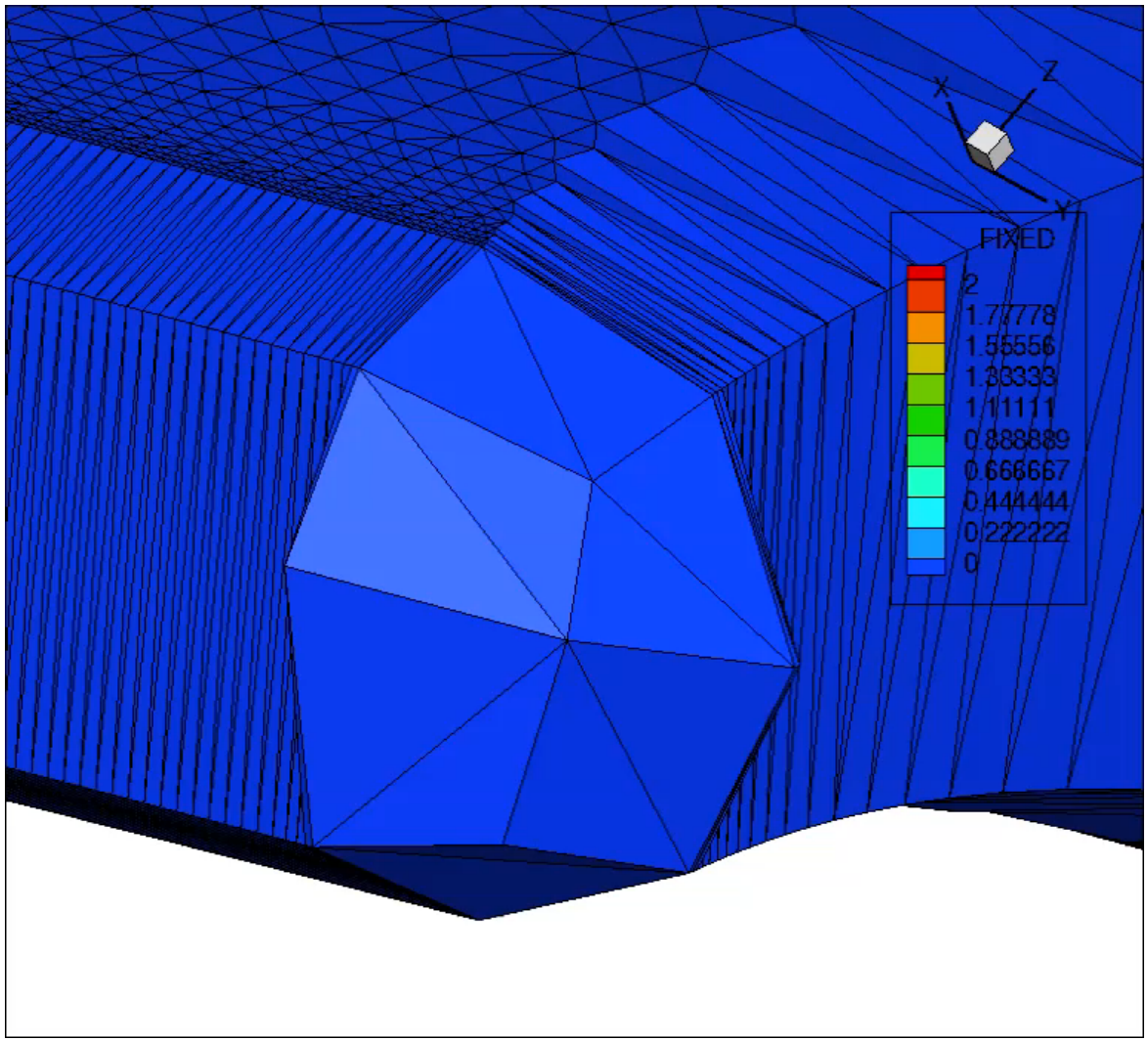
⇒ **Apply elastic smoothing algorithm, assuming each edge is an elastic spring with zero rest length, with constraints:**

1. Strand extremity remains on **Isosurface of Distance at L**
2. Strand remains within **visibility cone**



Spring stiffness such that nodes are at rest for very short strand lengths

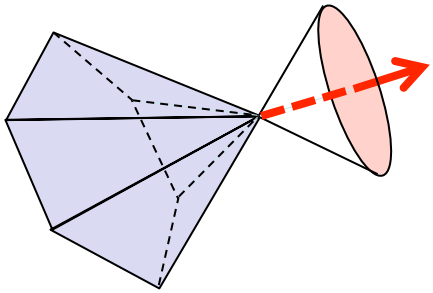
Elastic Smoothing on the Isosurface of Distance



- No constraint
- Visibility Constraint
- Angle Constraint

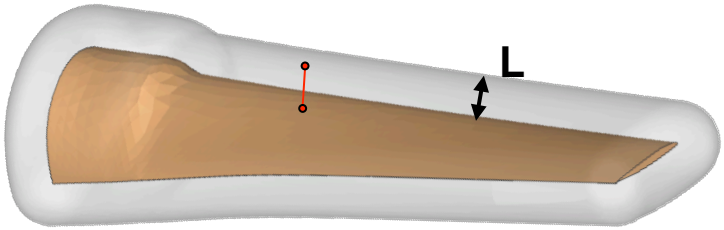
Meshing Method: Summary

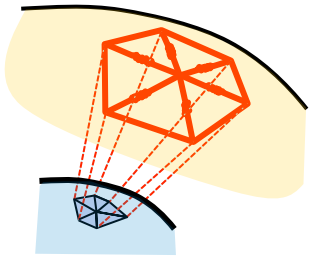
1. **Find Best Visibility direction for each surface node**

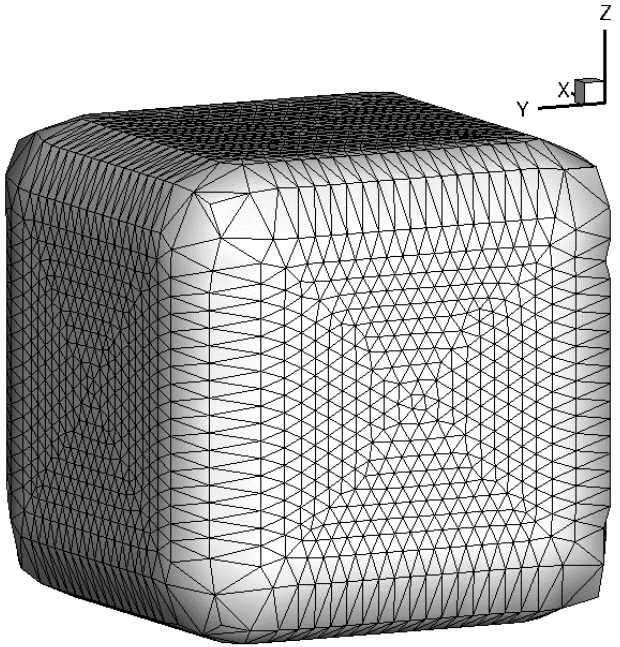


2. **Find Closest point on I_L for each surface node (CLOVIS method)**

   → Initial Strand mesh:
   CLOVIS if within cone of vis.,
   best visibility direction if outside

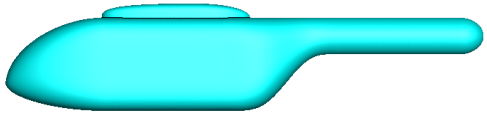






3. **Apply smoothing using spring analogy under constraints:**
   - Strand ends remain on I_L
   - Strands remain within their cone of visibility

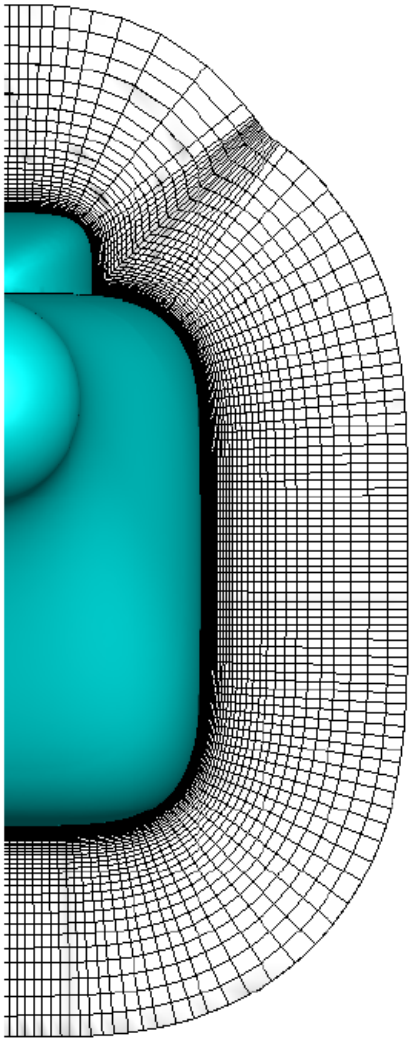
Mesh Quality Improvement
Robin Fuselage



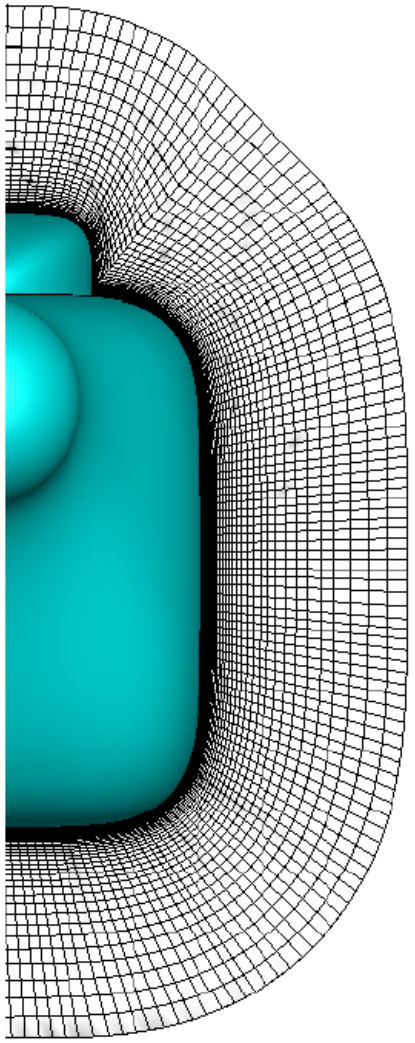
Side view



Rear view

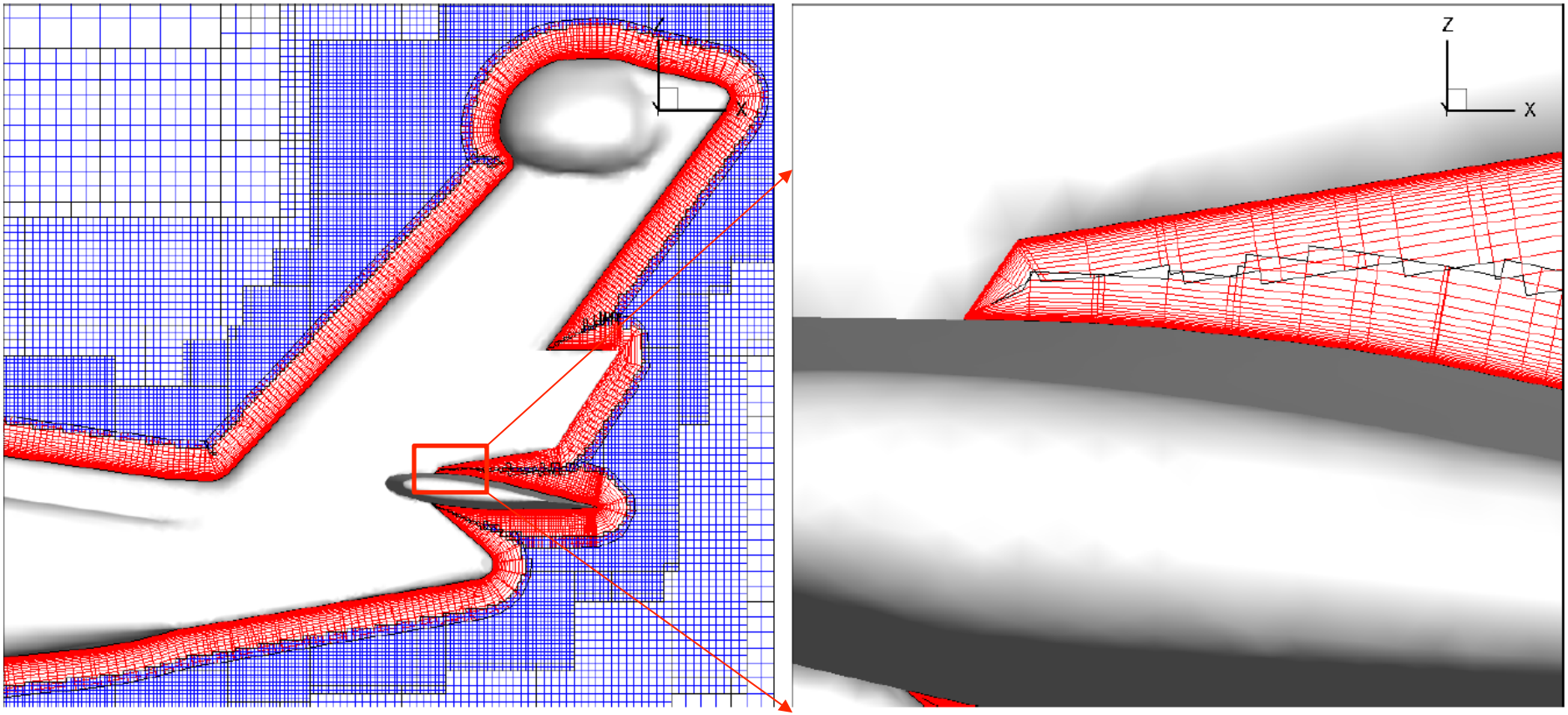


CLOVIS method



CLOVIS method + elastic smoothing

Complex Geometries with Self-Intersecting Meshes

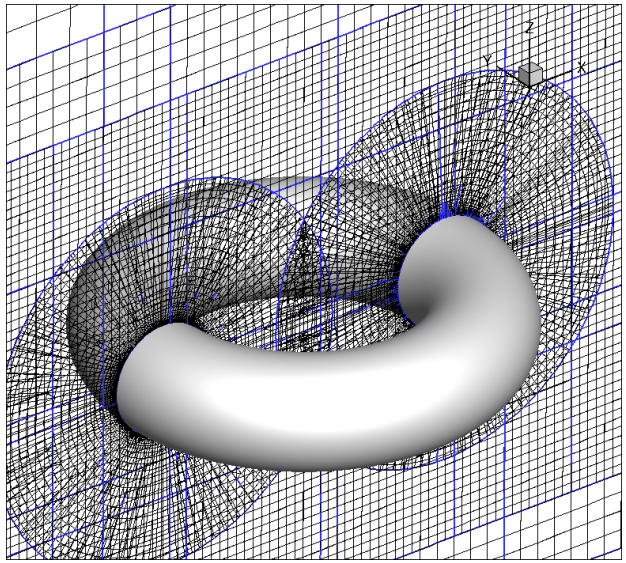


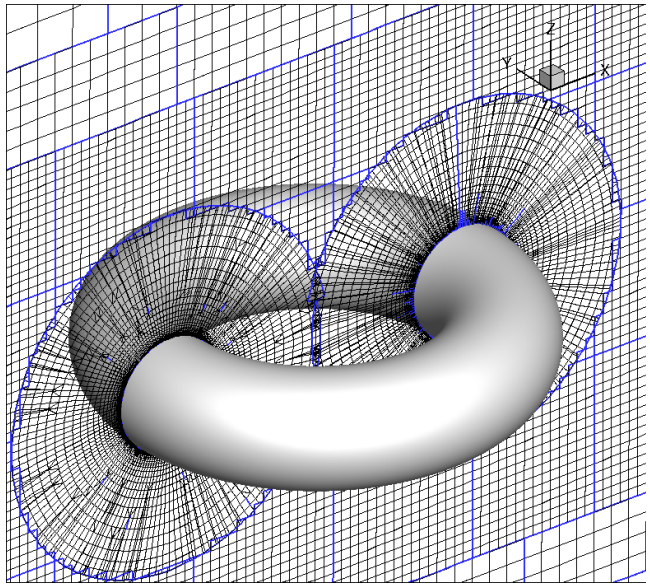
- Outer layer (single-level) built by simple extrusion from last level
- Complex geometry cannot be meshed without self-intersecting meshes.
  - Requires solution to strand collision to find which cells within the strand mesh solve, interpolate or are removed.
  - Key new technology developed in Helios V7

Domain Connectivity for Self-Intersecting Meshes

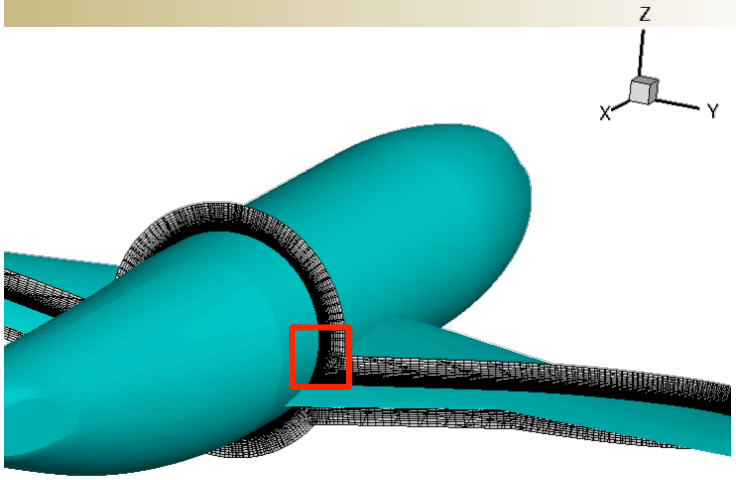
OSCAR (Overset Strand Cartesian Assembler)

- **Domain connectivity package specific for Strand/Cartesian grid systems and that uses compact grid description for scalable searches (Sitaraman 2012, 2013)**
- **Expanded to solve intra-mesh domain connectivity robustly**
- **ADT based search that performs rapid exclusions and exact face-edge intersection checks.**
- **Complex logic for clearing intersections**
- **Additional challenges (orphan creation)**

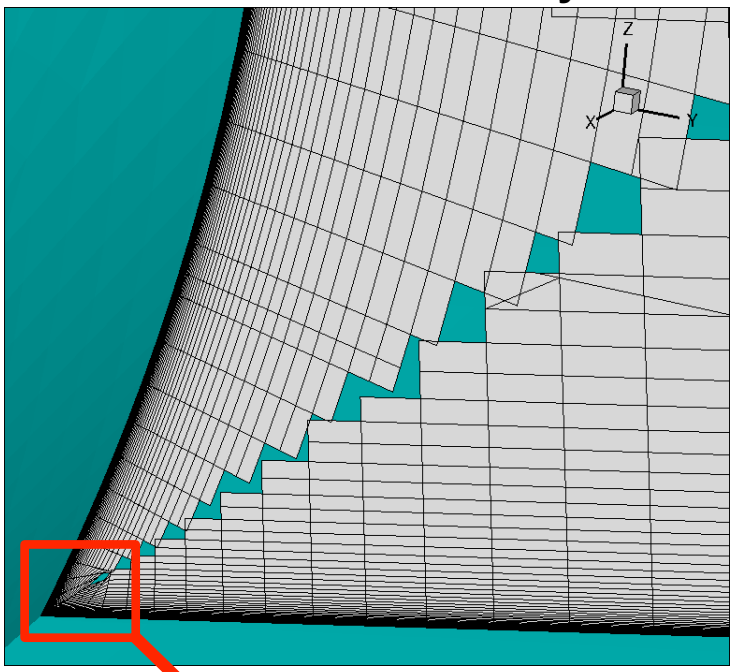


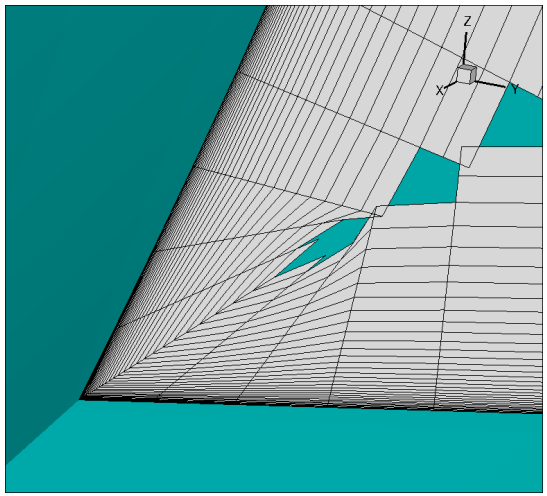


Intra-Mesh Domain Connectivity

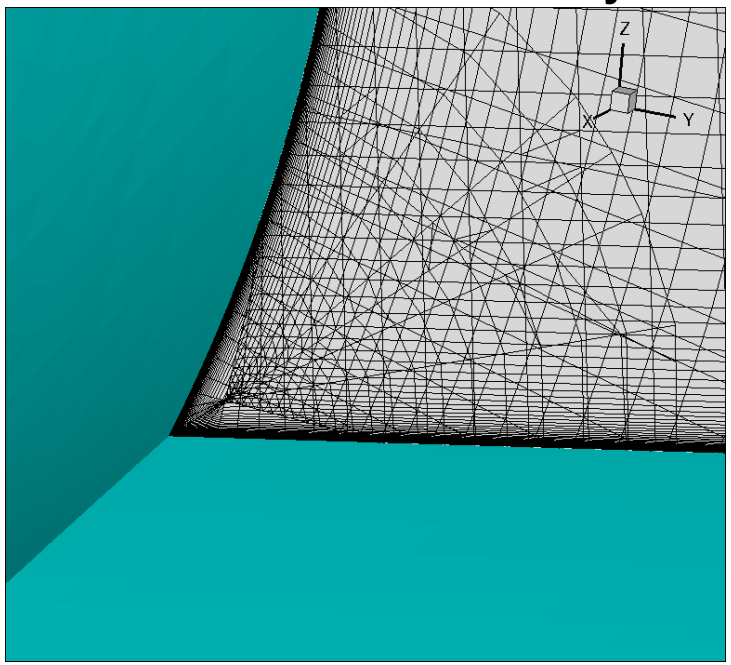


After domain connectivity

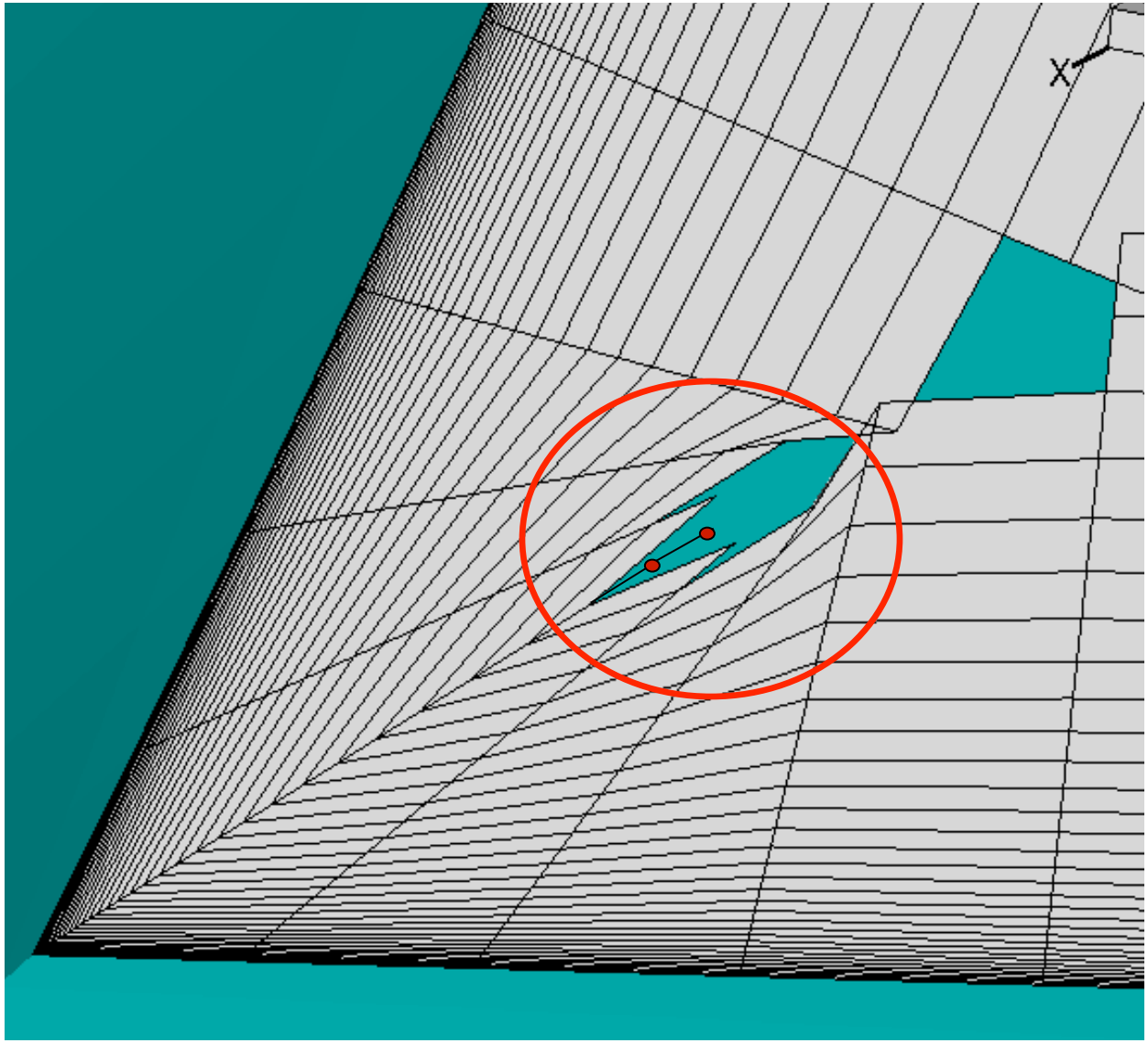




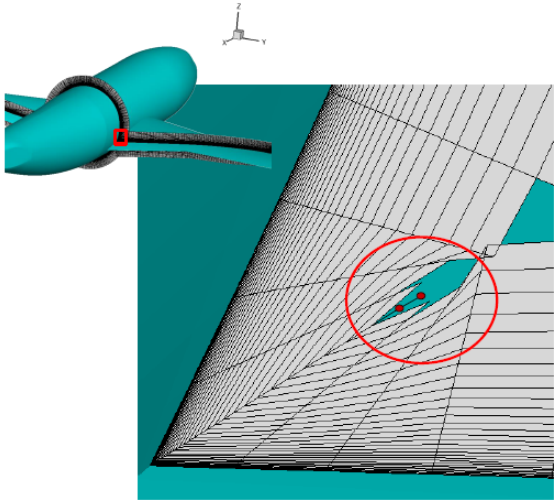
Before domain connectivity



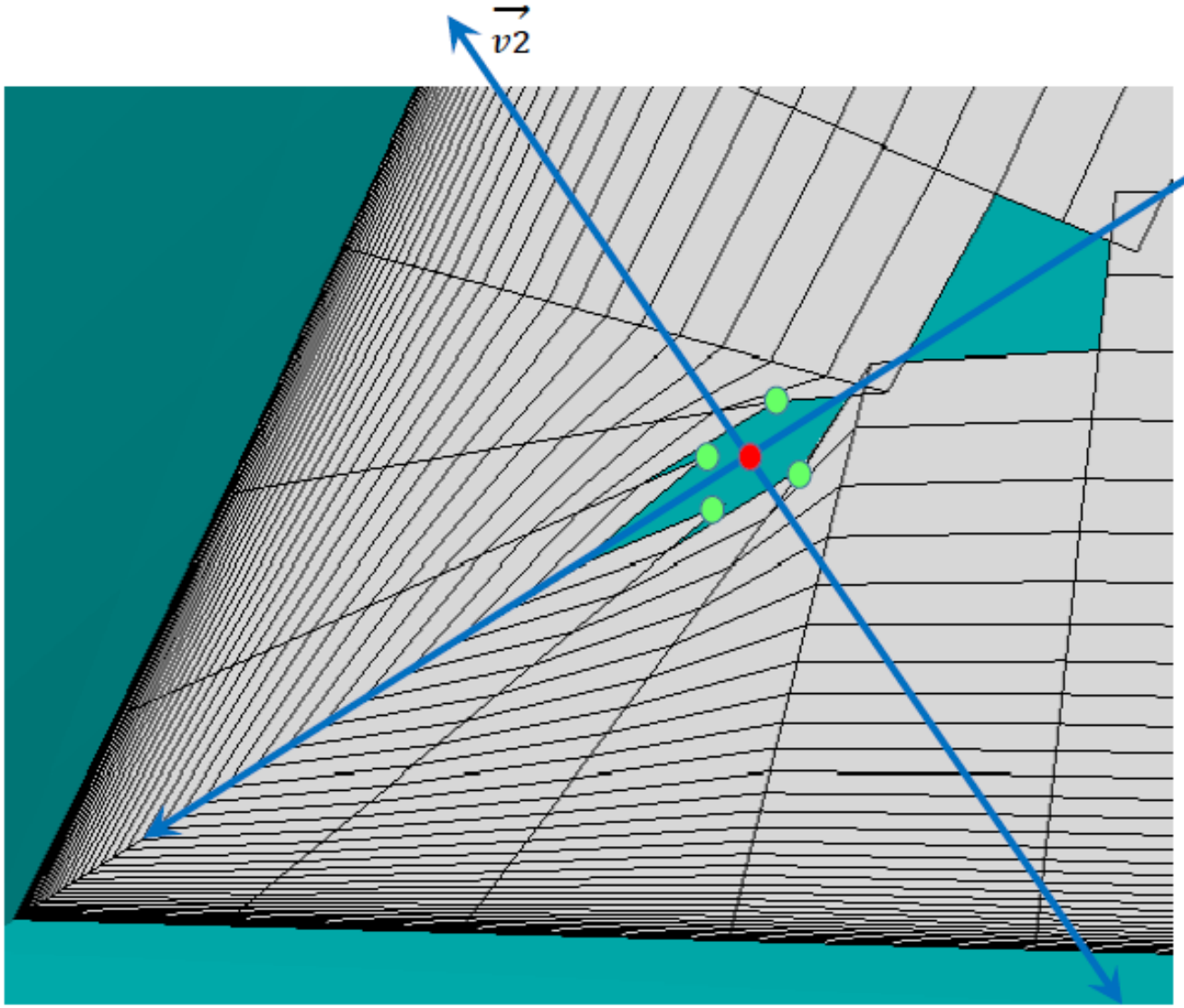
Issues near shallow-angle strand intersections



- Fringe points that cannot find any valid donor (i.e. a cell with field points)
- Cartesian donors could be facilitated if the AMR mesh can penetrate all the way, but it is undesirable because of the anisotropy



Solution to orphan points



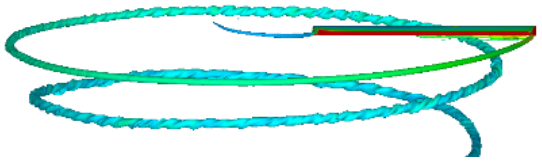
- Vectors that form octants
- Receptor Node
- Donor Node(s)
  - Found through octant constrained search process.
  - Each donor node is the closet point to the receptor node within each octant.

Least-square interpolation using point cloud (2nd order accurate)

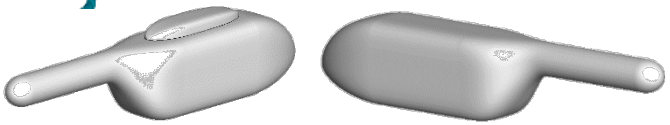
Results

Overview

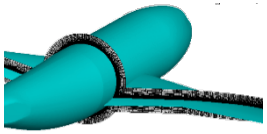
1. **Hovering 1-bladed rotor**
   - Simple blade geometry



2. **Simple Fuselages (Robin w and w/o pylon)**



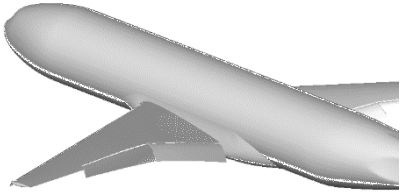
3. **Moderately complex fuselage (DLR-F6)**



4. **UH-60 Fuselage**
   - With and without geom. details



5. **High-Lift CRM**
   - multiple complex meshes body/flap/slat
   - Both inter-mesh and intra-mesh connectivity



6. **Rotor + Fuselage in Trimmed flight**
   - Elastic blades (coupling with CSD)
   - Comparison with flight test data

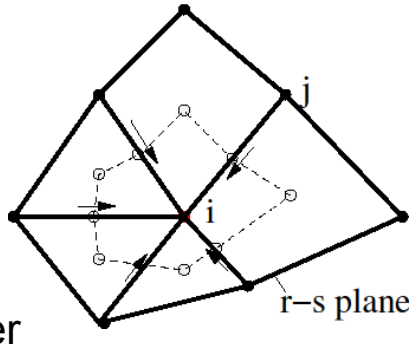


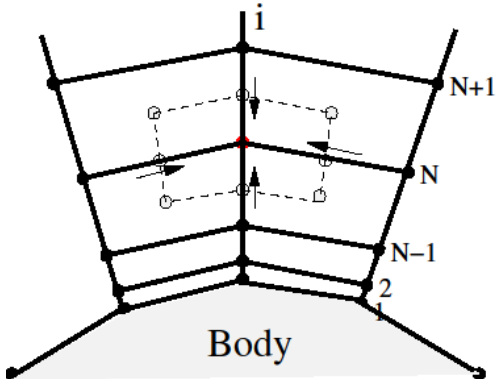
All volume meshes used for the results are auto generated

Solution process in Helios

Near-body strand solver (mStrand)

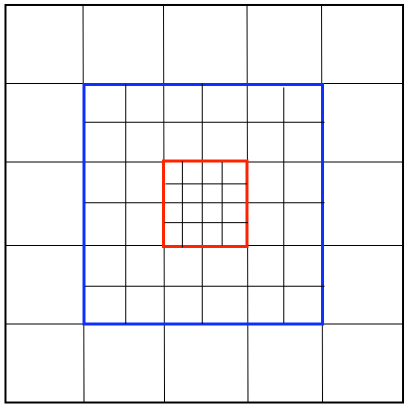
- Fully parallel 2nd order vertex centered FV solver
- Surface grid can be quads or/and triangles
- General prismatic mesh in normal direction
- 3rd order MUSCL reconstruction + Riemann solver
- 2nd order full Navier-Stokes term
- 1st order implementation of SA turbulence model
- Moving/deforming meshes with ALE and discrete GCL
- Can handle multiple strands/node
- Linear solver using preconditioned GMRES





Off-body adaptive Cartesian Solver (SAMCart)

- Parallel mesh adaptive capability by SAMRAI library
- Each grid block solved using CART
- Higher order central difference scheme
  - ➔ 6th order with 5th order dissipation for inviscid term
  - ➔ 4th order for viscous terms
- SA turbulence model with DES capability
- LUSGS or diagonalized ADI implicit operators

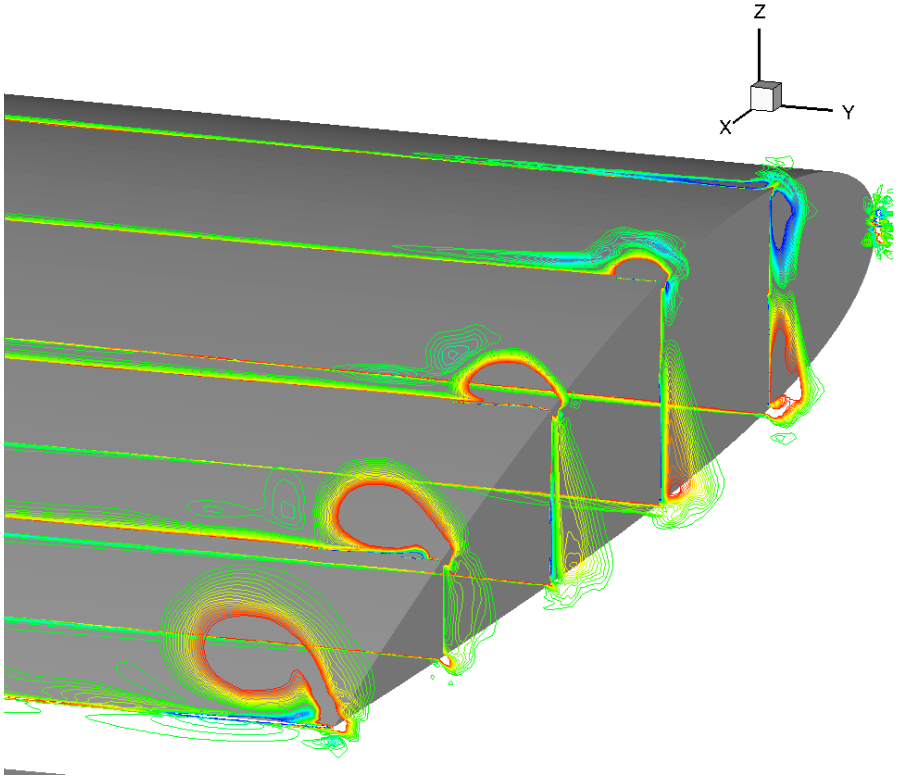


Berger-Colella style block structured AMR

Case 1: Hovering Rotor Wake Validation

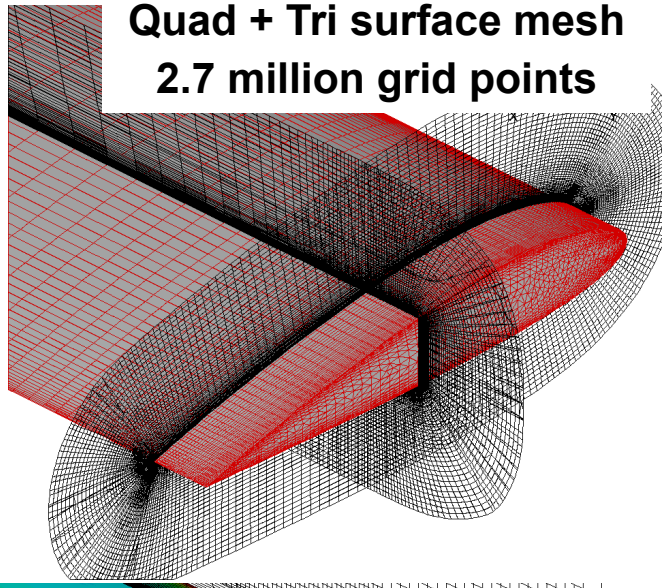
One bladed rotor tested by Martin et al.

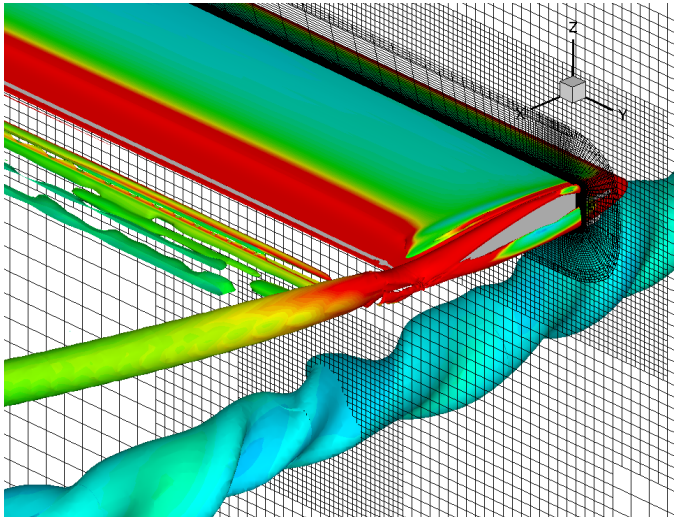
- Aspect ratio → 9.12
- Tip Mach → 0.26
- Tip Re number → 272,000
- Collective → 4.5°



Vorticity Magnitude

Quad + Tri surface mesh
2.7 million grid points



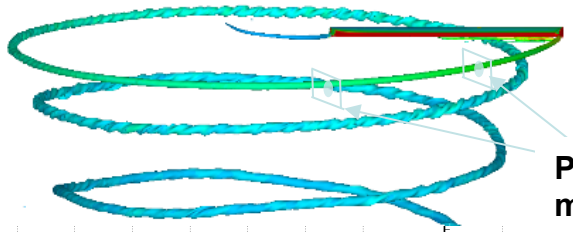


Iso-surface of Q-criterion colored by vorticity magnitude

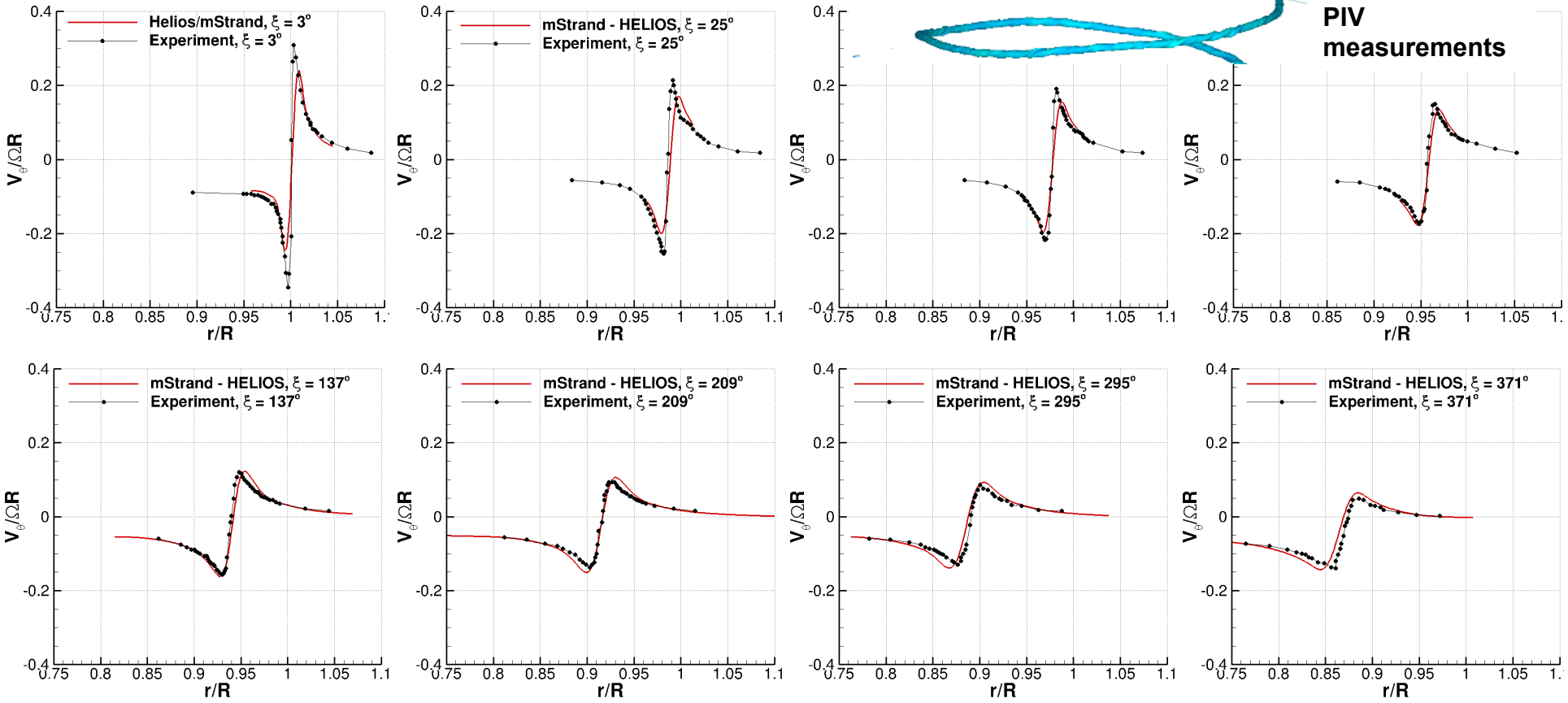
Case 1: Hovering Rotor Wake Validation

One bladed rotor tested by Martin et al.

Swirl velocity profile across the center of vortex

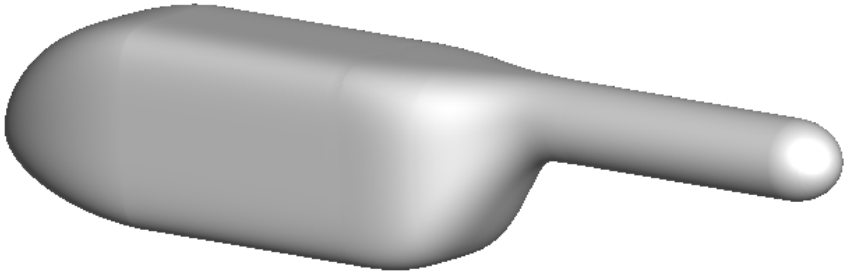


PIV measurements



- **Vortex strength preserved well for long time**
- **Wake contraction and vortex dissipation rate predicted accurately**

Robin Fuselage no pylon

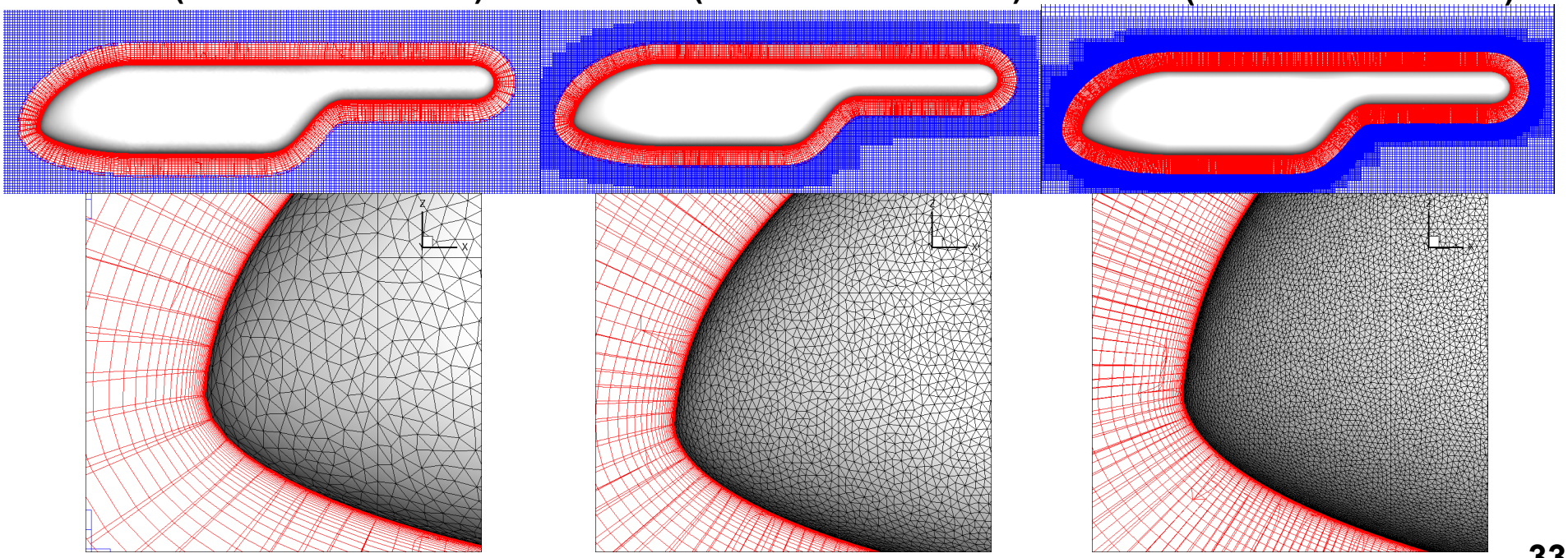


- Generic analytically defined fuselage
- Simple geometry, with gentle convexity and concavity
- can generate strand meshes to arbitrary distance away from the body

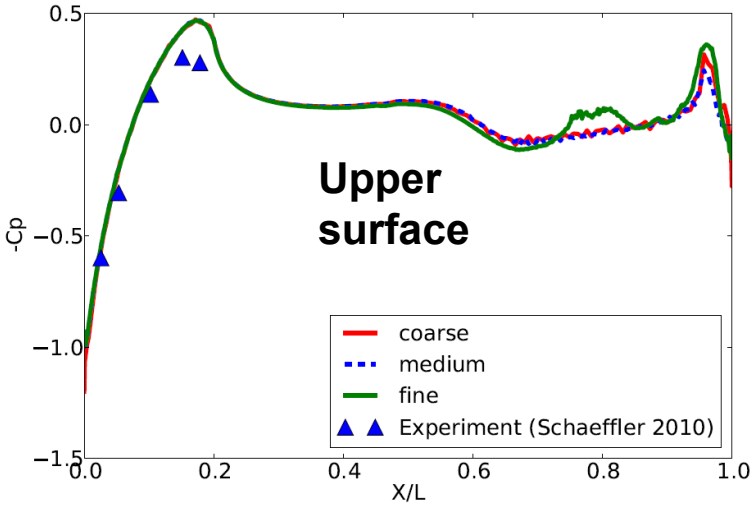
Coarse (0.6M NB + 3M OB)

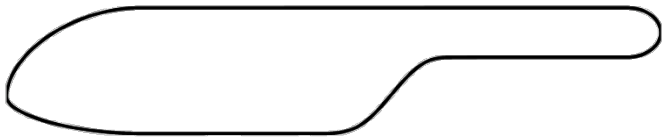
Medium (2.4 M NB + 8 M OB)

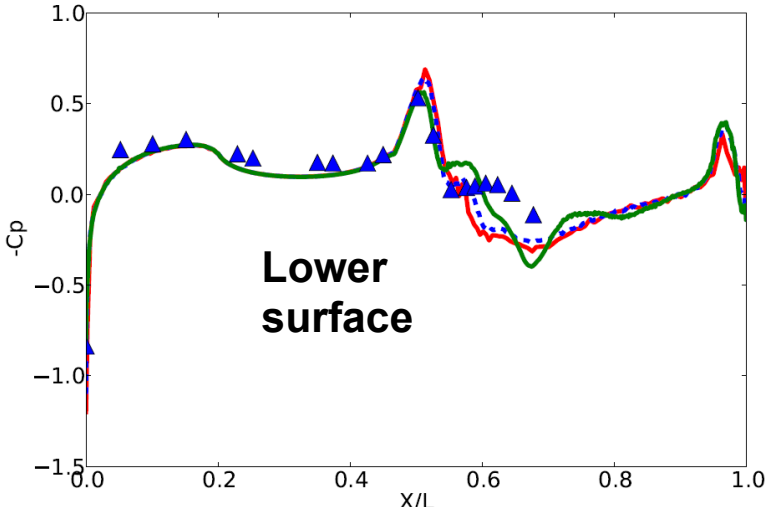
Fine (10M NB + 35 M OB)

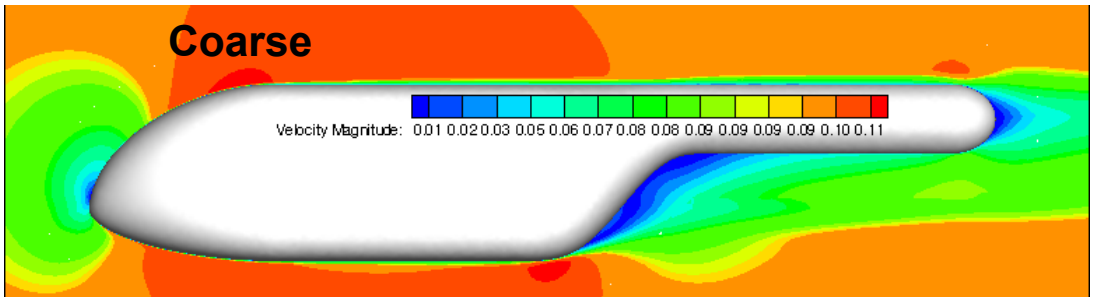


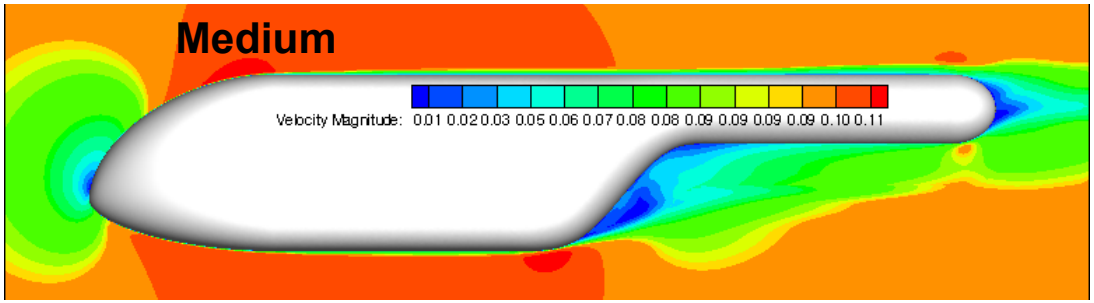
Robin Fuselage no pylon

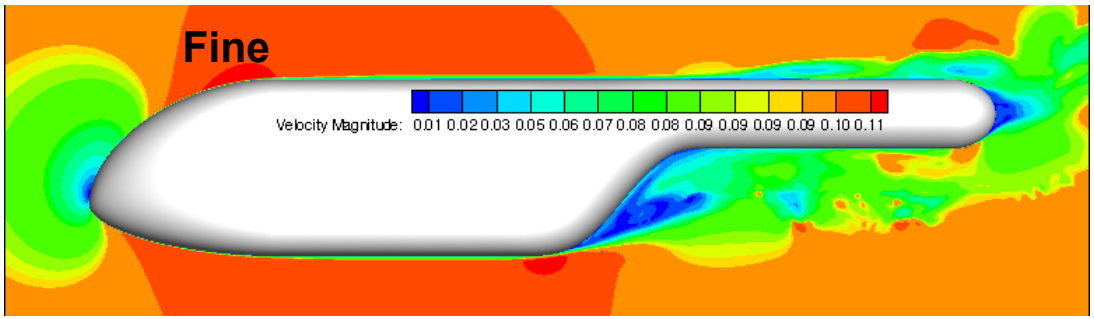






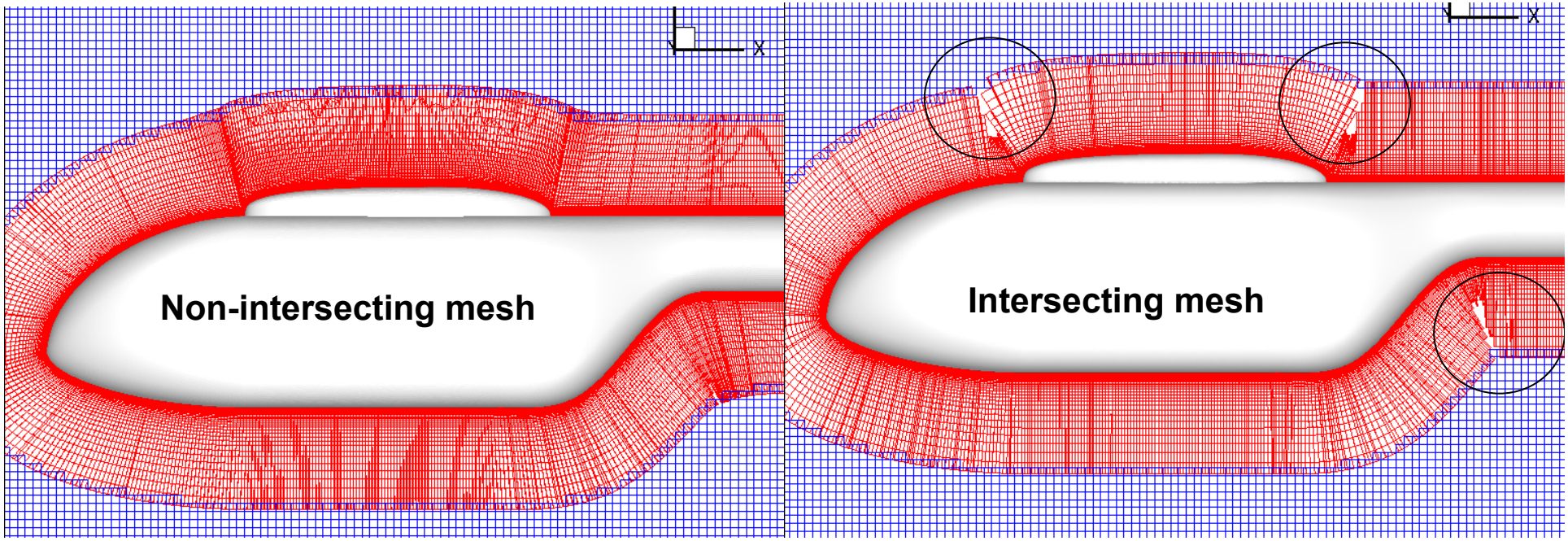






M=0.1, dt=5e-5, time-accurate, unsteady residual converged to 6 orders every time step

Robin Fuselage with pylon

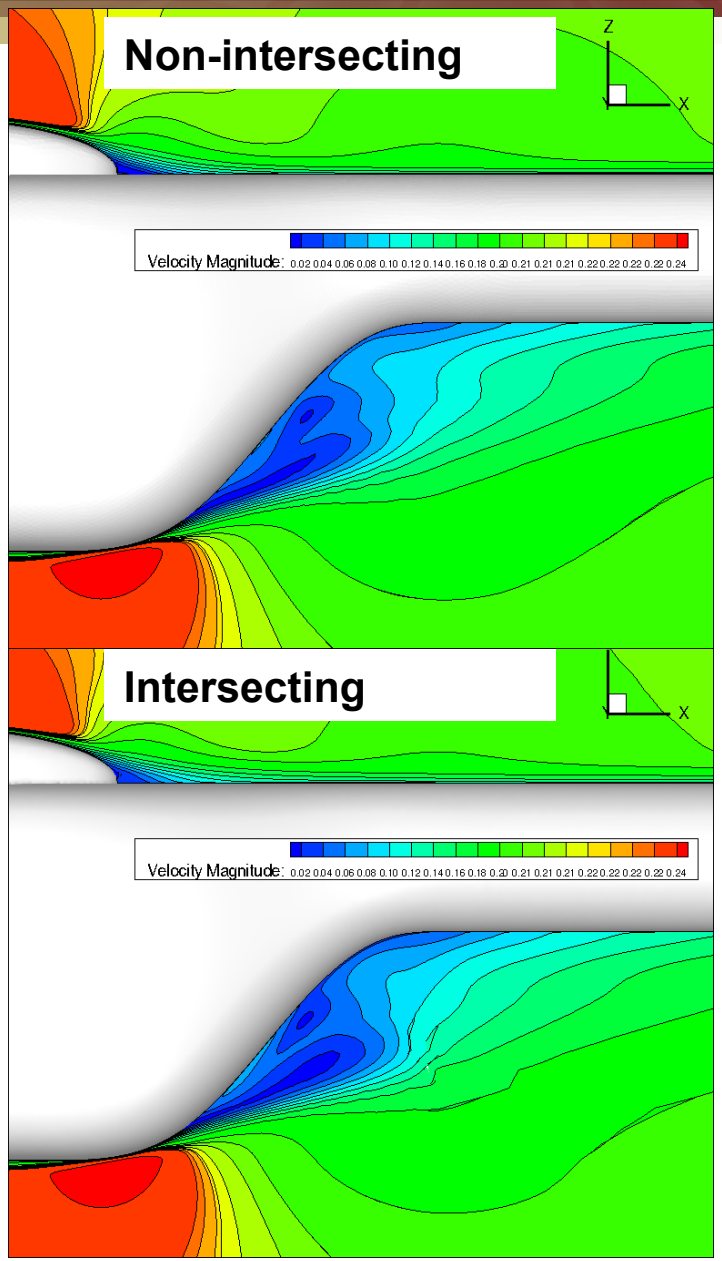


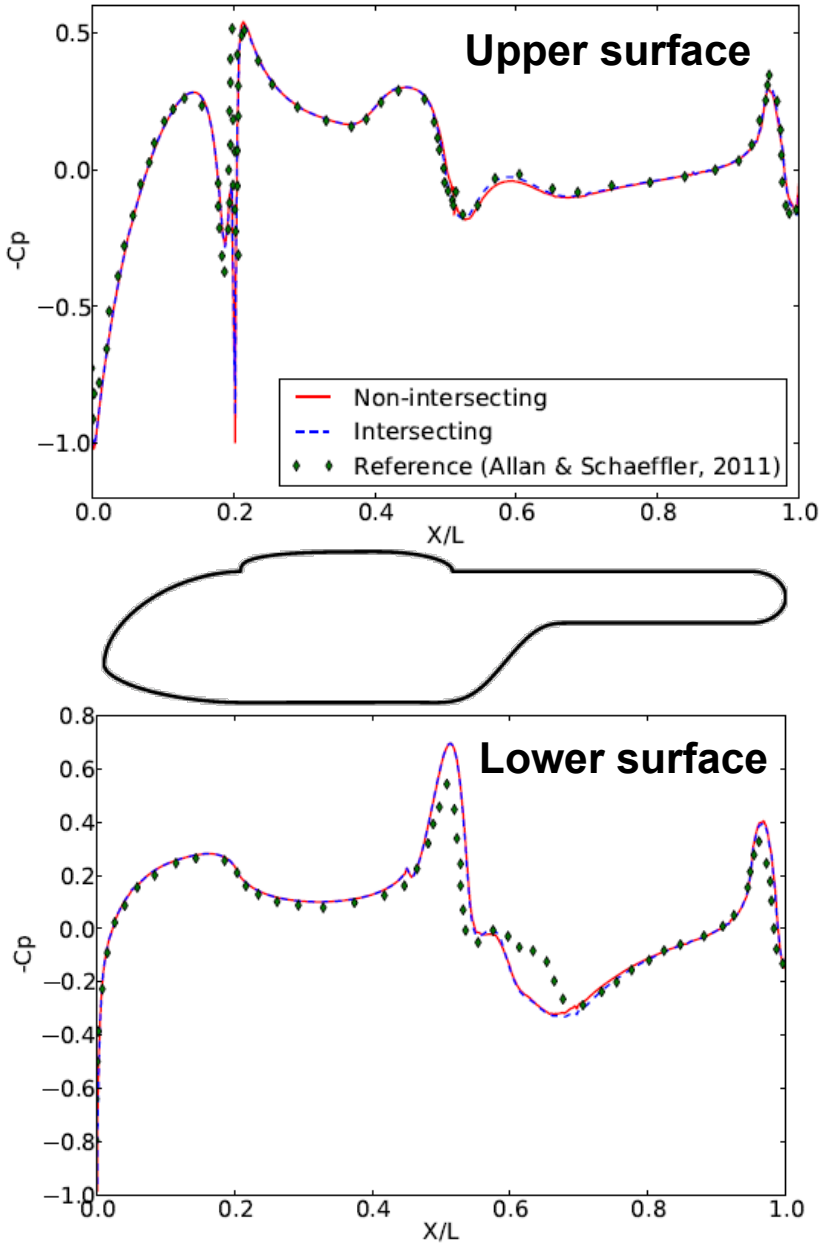
Intersecting mesh generated by extending a non-intersecting mesh at a smaller distance from the wall along the strand-vector

Self-intersections resolved at run-time using OSCAR domain connectivity

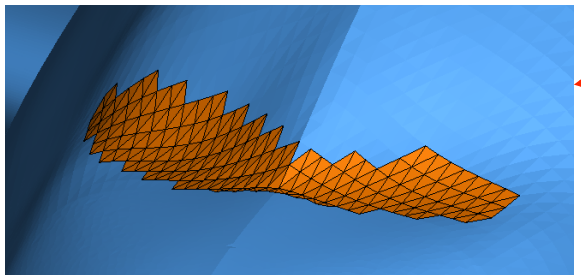
Objective: Compare solutions for the same geometry using intersecting and non-intersecting meshes

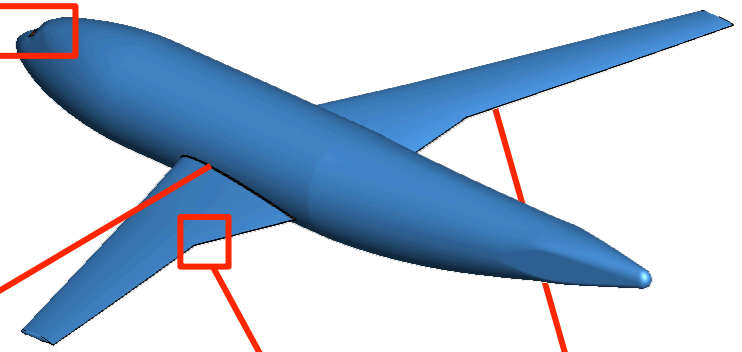
Robin Fuselage with pylon

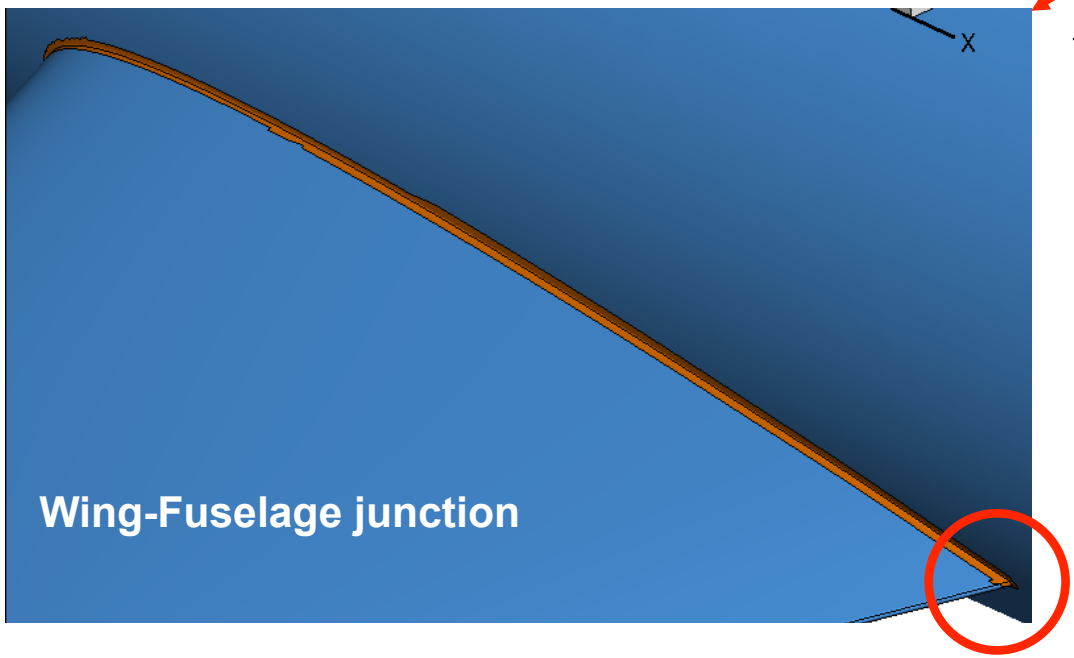


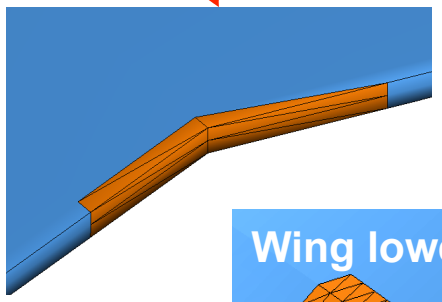


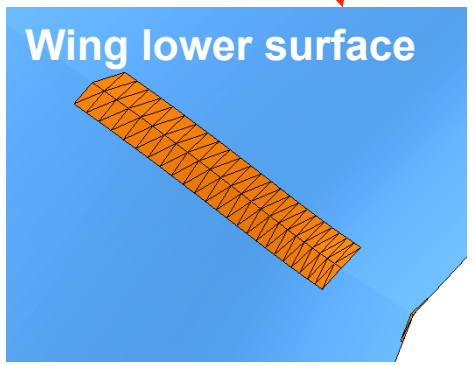
DLR-F6 Fuselage From DPW Series









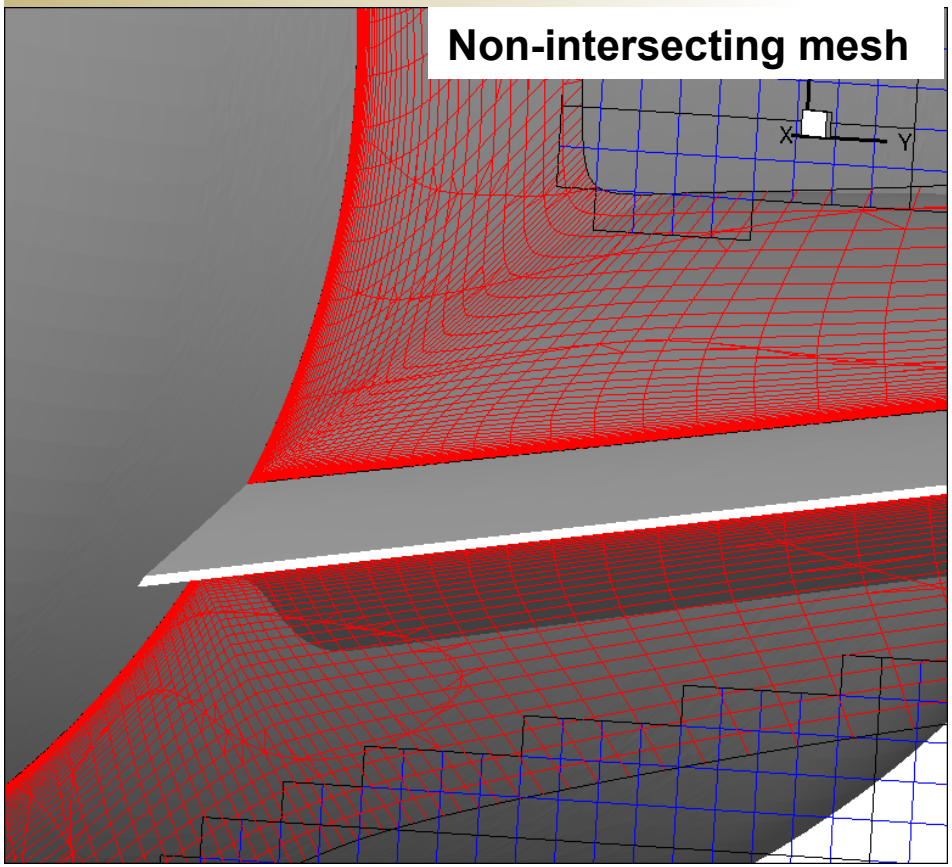


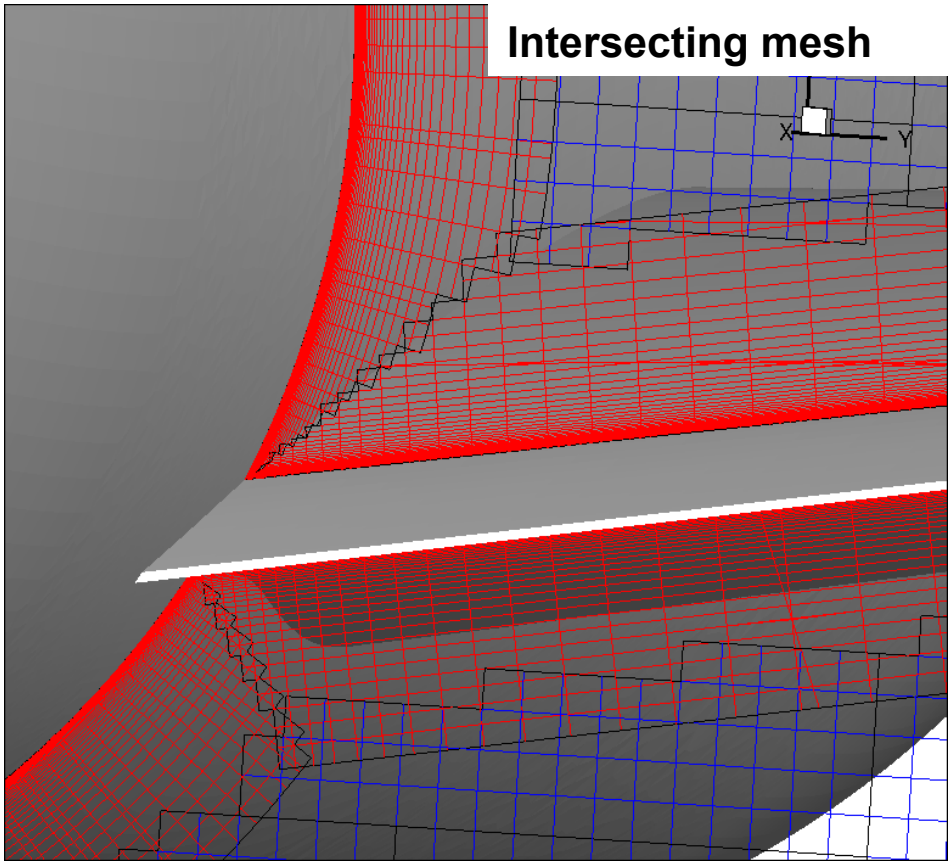
More complex than ROBIN fuselage

Concavities at wing fuselage junction

Concave-convex intersection at the trailing edge

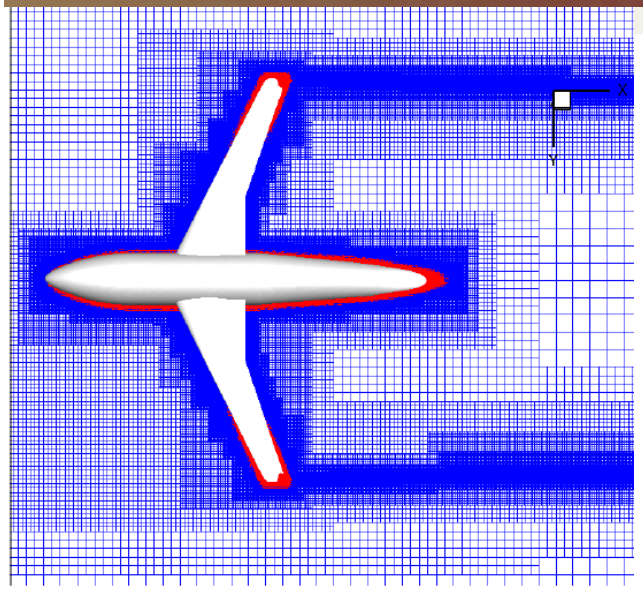
DLR-F6 Fuselage Mesh generation



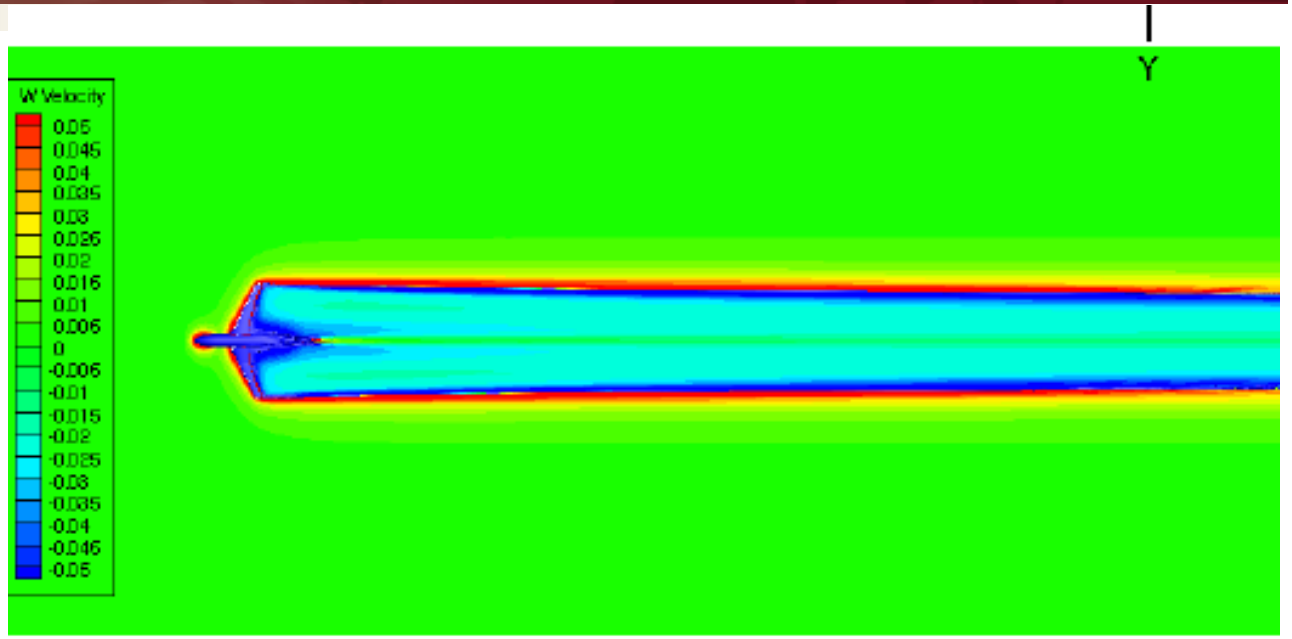


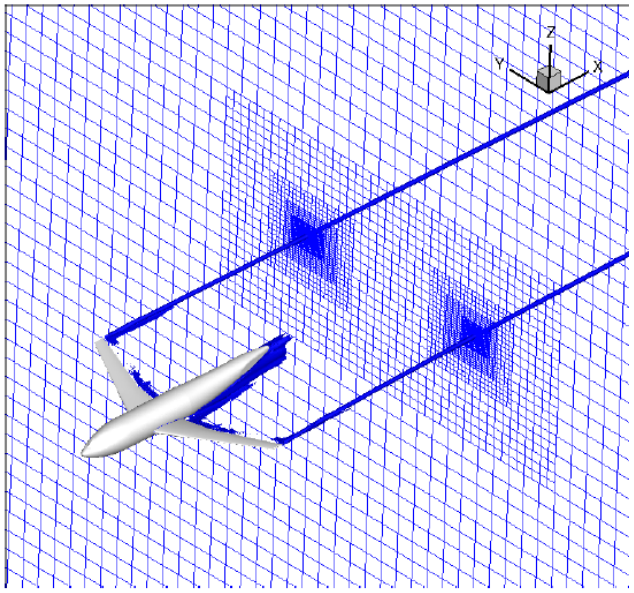
- Strand meshing technique can generate valid multi-level strand meshes up to 1/6th of wing mean chord. Most commercial mesh generators can only generate prismatic meshes for this geometry to 1/20th of the mean-chord.
- Intersecting meshes generated by extending a 1/100th chord mesh to 1/6th chord

DLR-F6 fuselage Flow solution



(a) Mesh System used for DLR-F6 computation

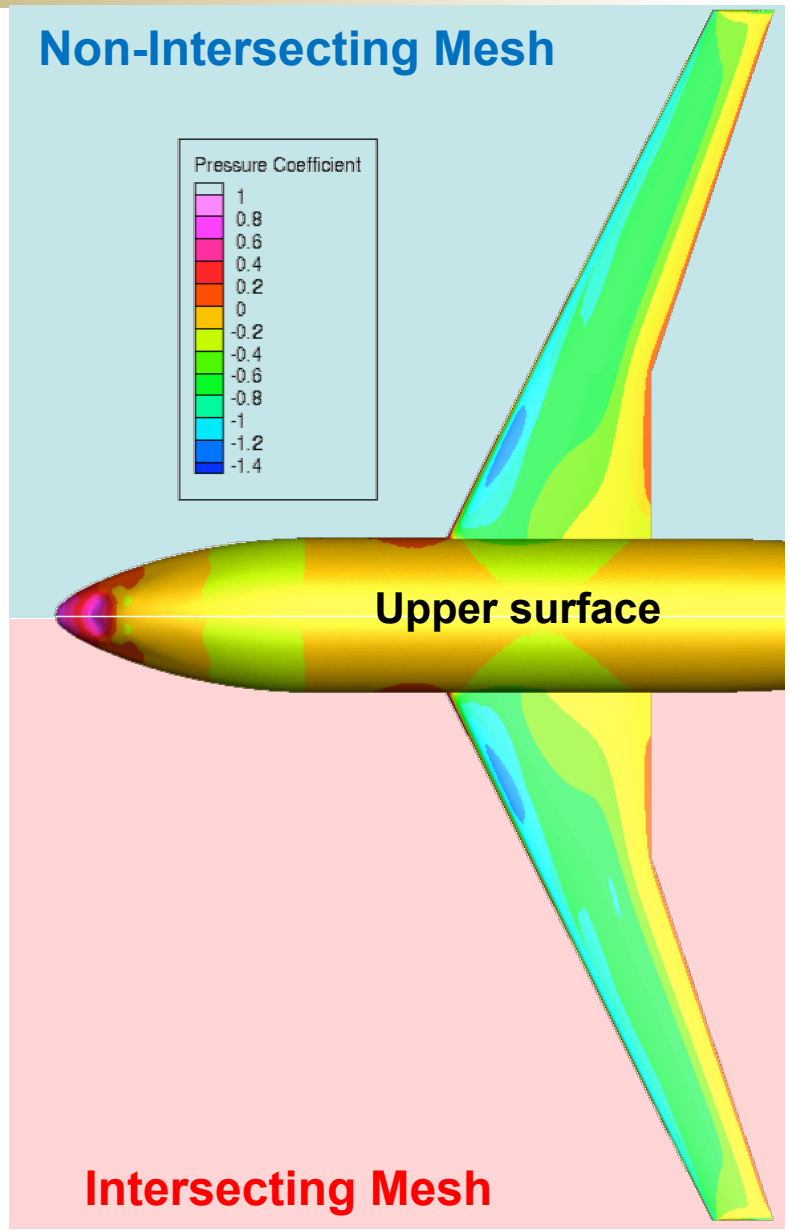


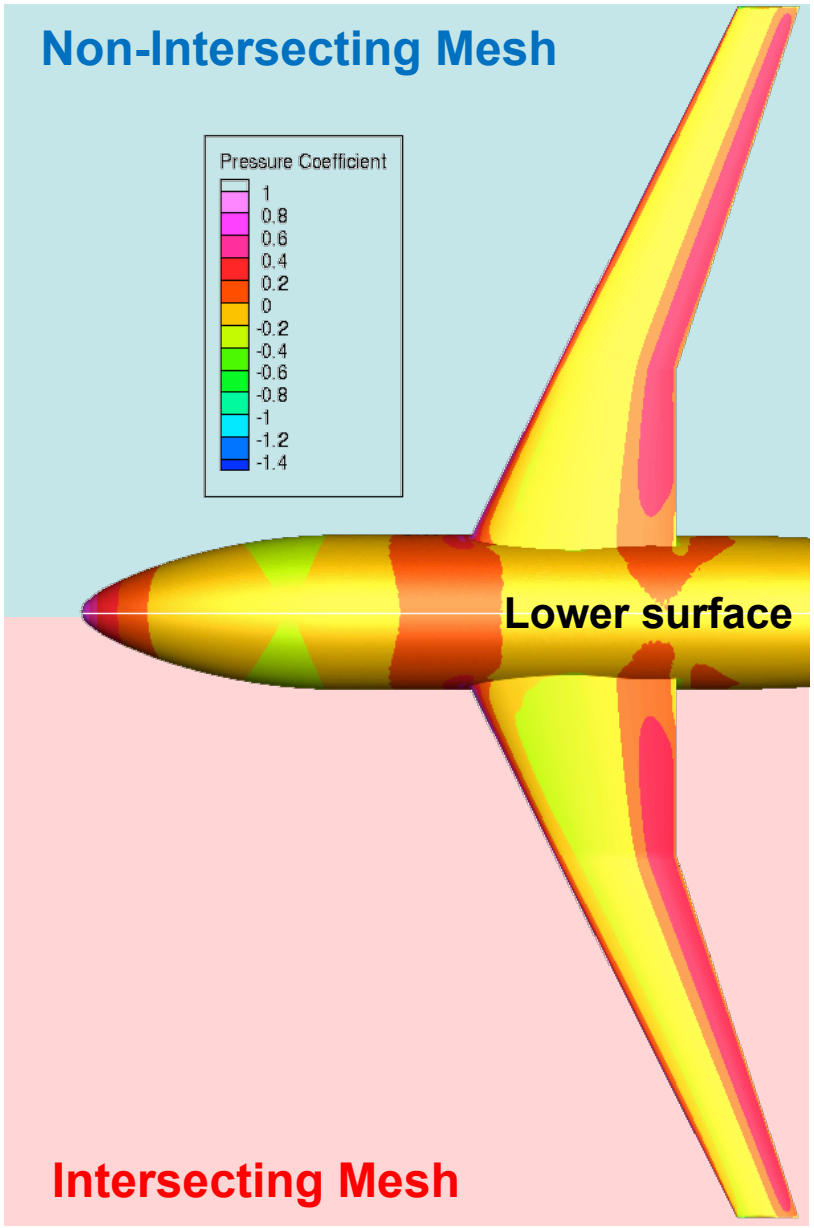


Solution based AMR tracks wake to 10 aircraft lengths

Drag converges to within 1 count of final in about 5000 iterations for both non-intersecting and intersecting meshes

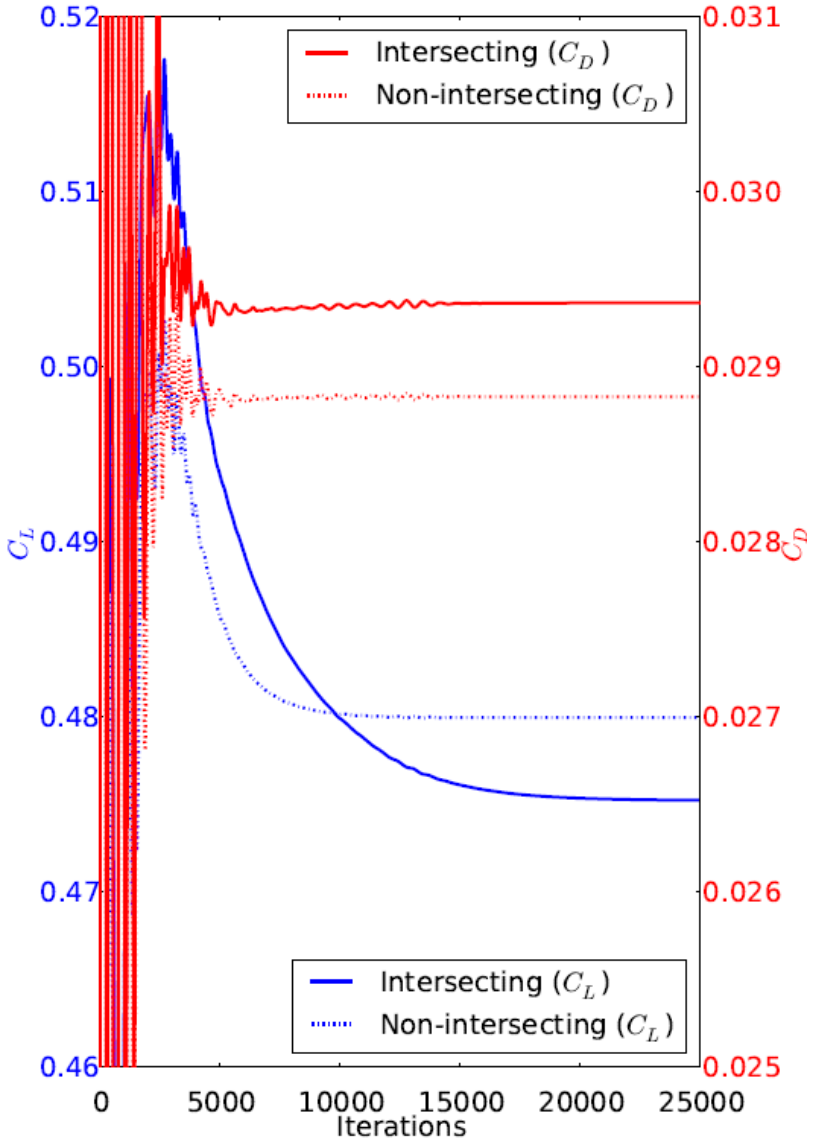
DLR-F6 Fuselage Pressure contours

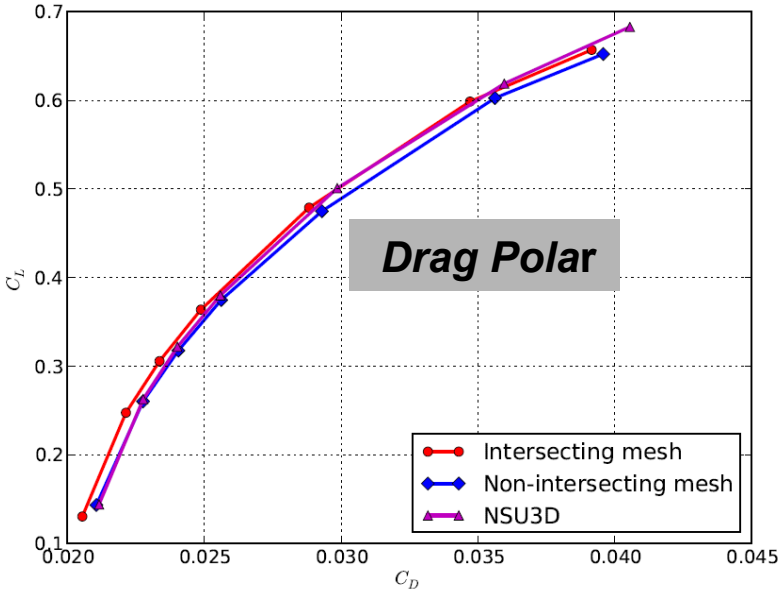




DLR-F6 Fuselage CL/CD and Polar

CL/CD evolution for α=0





CL/CD at α=0 from DPW-II/III

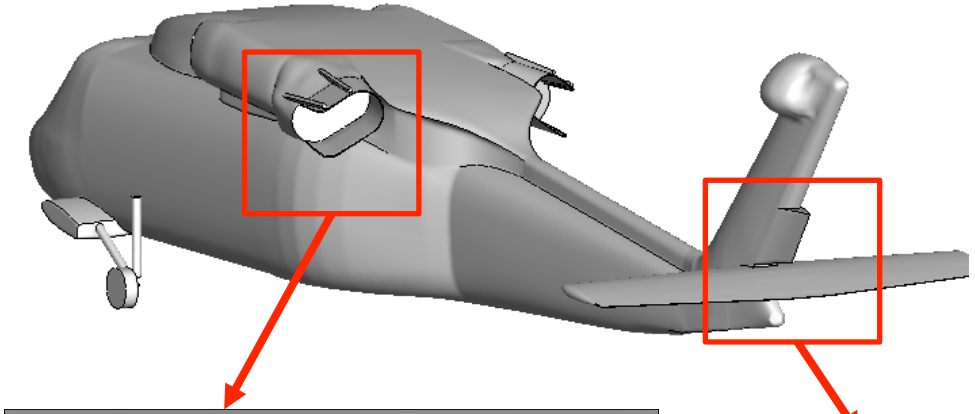
Code	C_L	C_D
FUN3D[49]	0.477	0.02766
NSU3D[38]	0.507	0.02942
OVERFLOW[37]	0.494	0.02745
CFL3D Overset[52]	0.483	0.0272
CFD++[51]	0.483	0.02848
Tau[50]	0.491	0.02783
HELIOS (intersecting)	0.479	0.0288
HELIOS (Non-intersecting)	0.475	0.0293

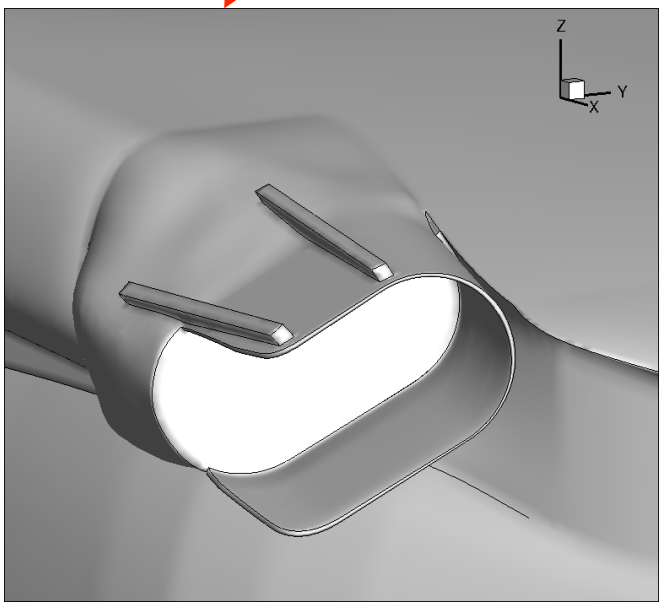
Case 2: UH-60A Fuselage

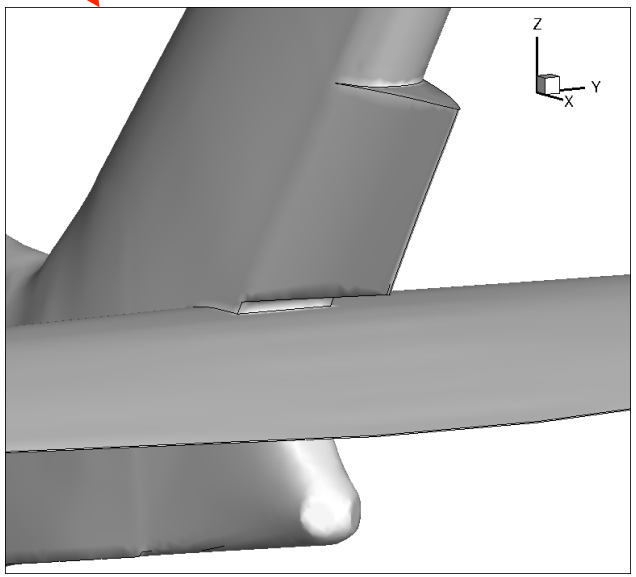
Real Fuselage test case with many geometric complexities

Non-intersecting mesh cannot be produced for more than 0.5 inch strand length from surface.
blade chord = 20.76 inches

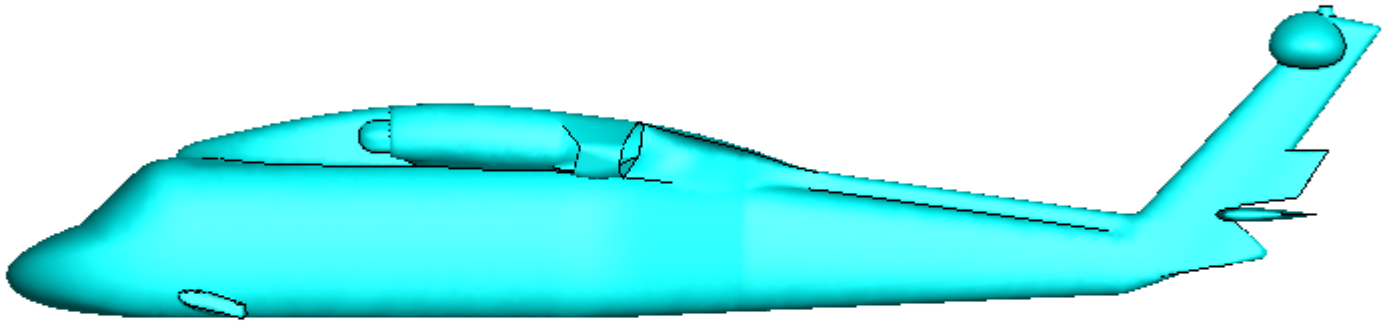
Intersecting mesh generated by extending a valid strand mesh at 0.5 inch to a distance of 10 inches.



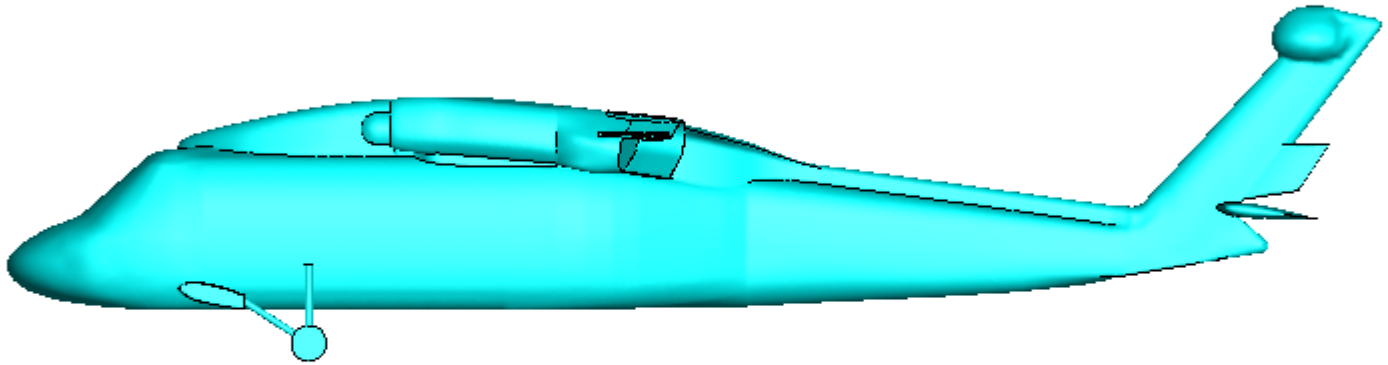




UH-60 Configurations



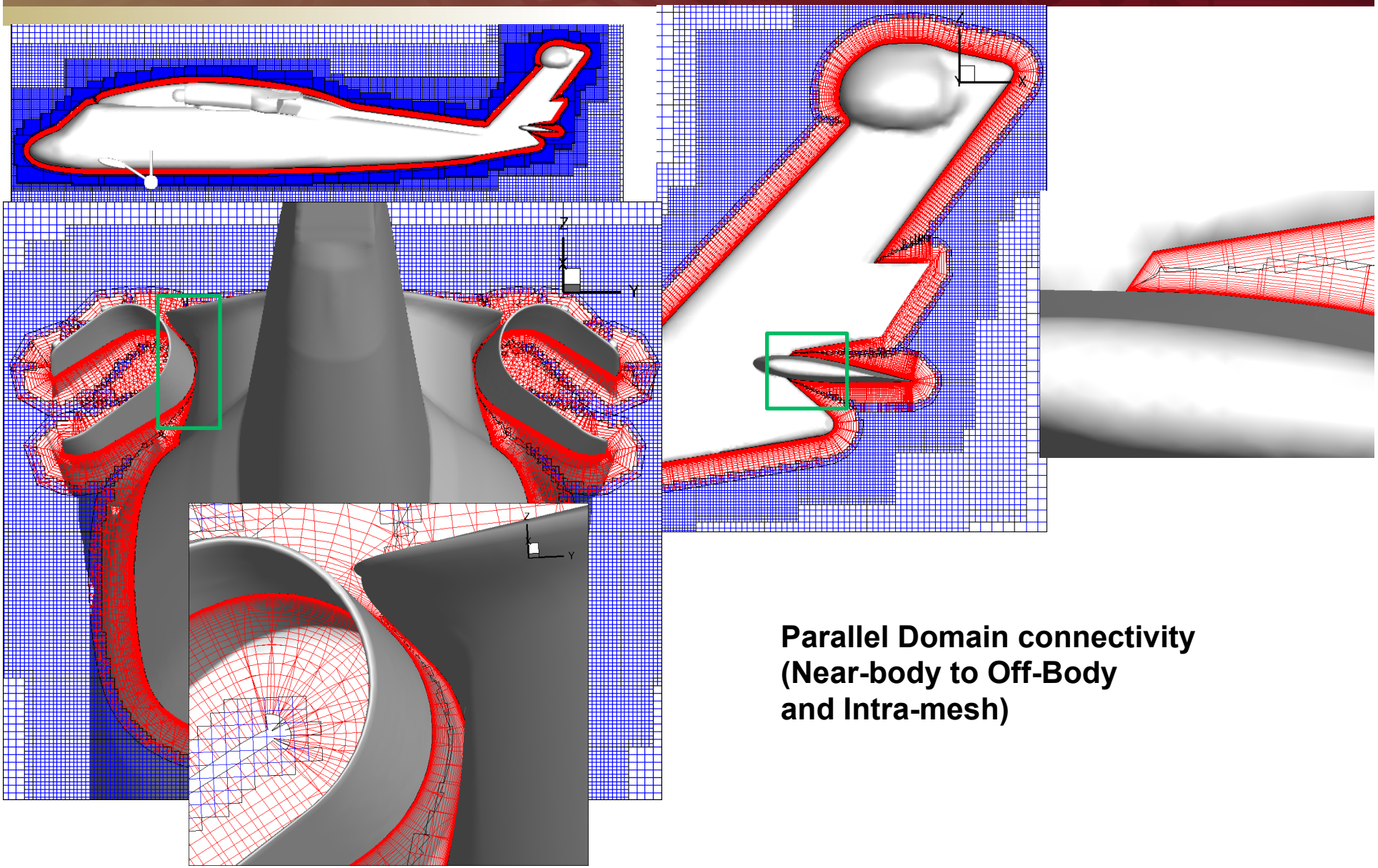
Clean



Strut:
- Landing struts
- IR suppressors
- Other small details near the engine inlet

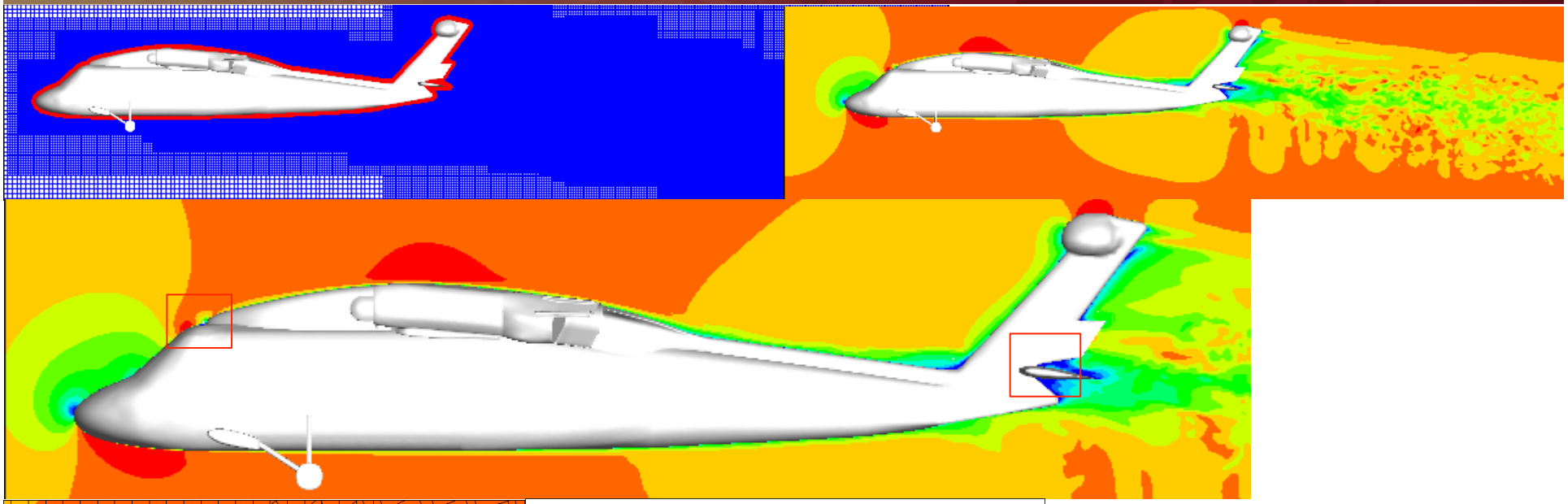
Questions:
- Can we mesh and solve both configurations using Strand/Cartesian approach?
- Can we predict the drag increment in the Strut configuration compared to clean?

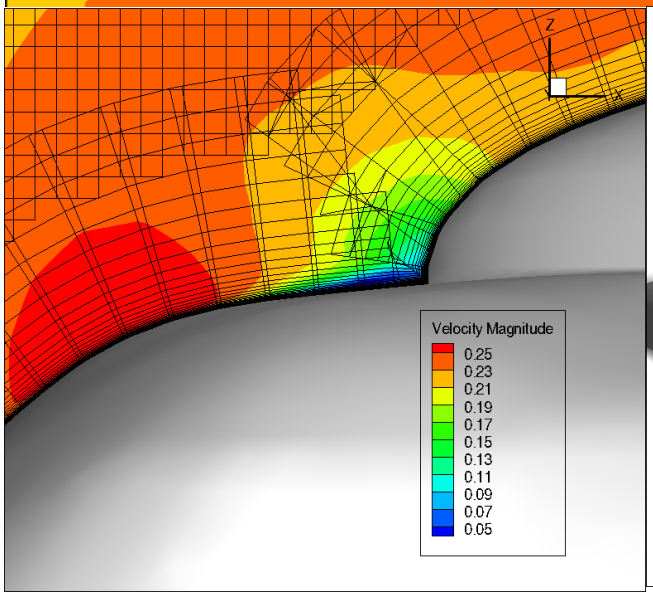
UH-60A Fuselage Domain Connectivity

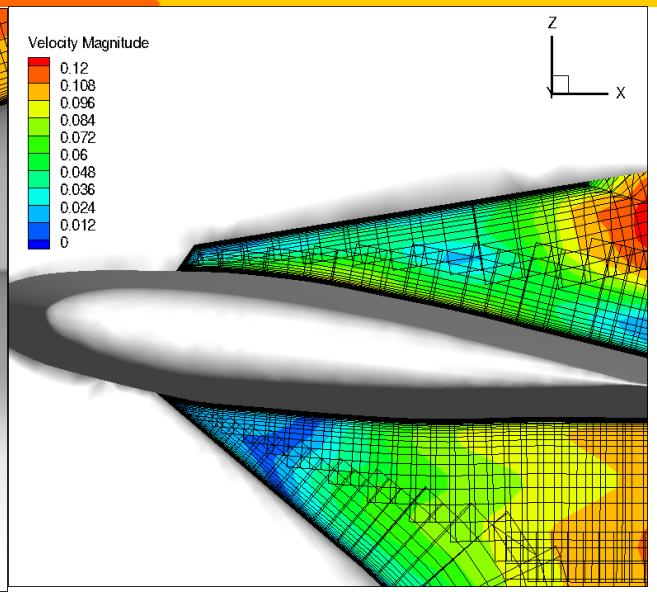


Parallel Domain connectivity (Near-body to Off-Body and Intra-mesh)

UH-60A Flow Solution: Fuselage with Struts





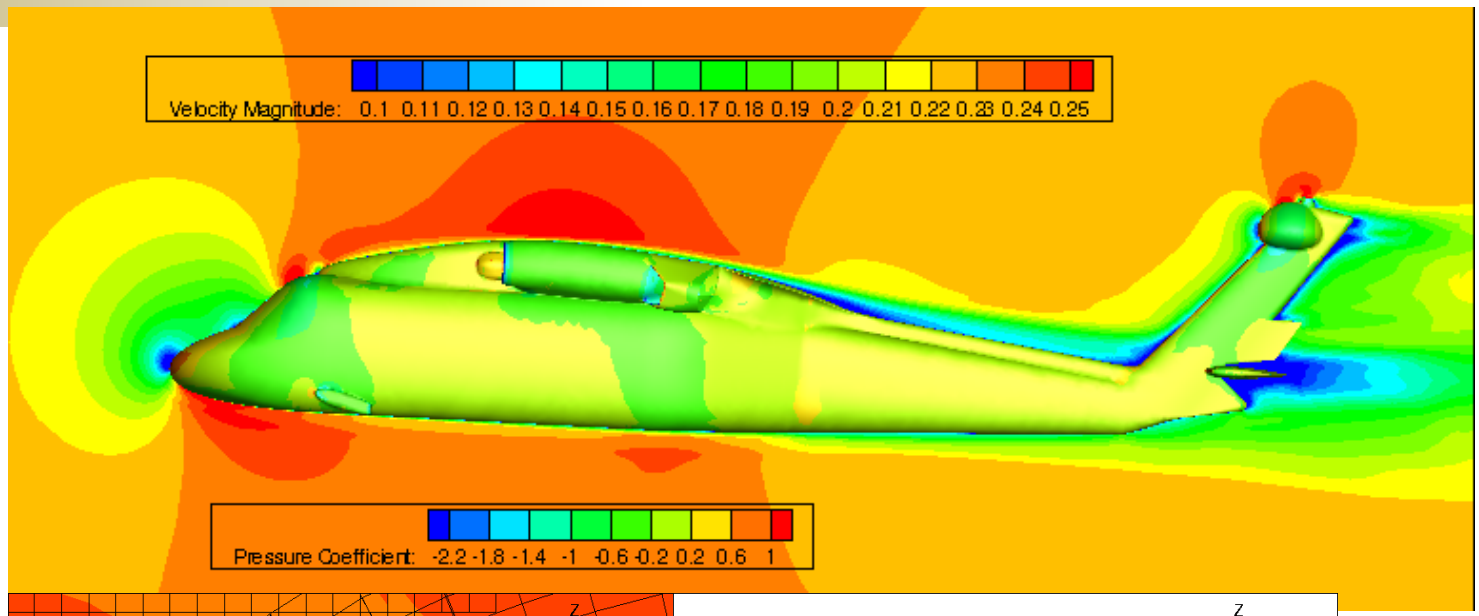


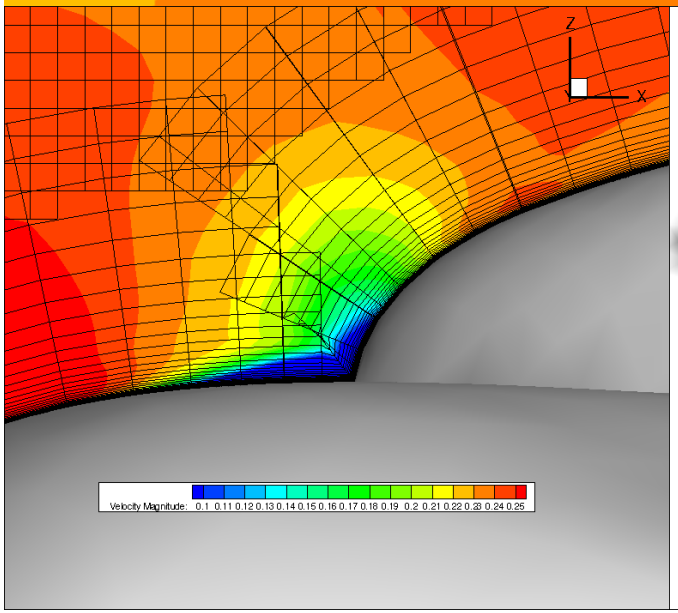
High speed forward flight

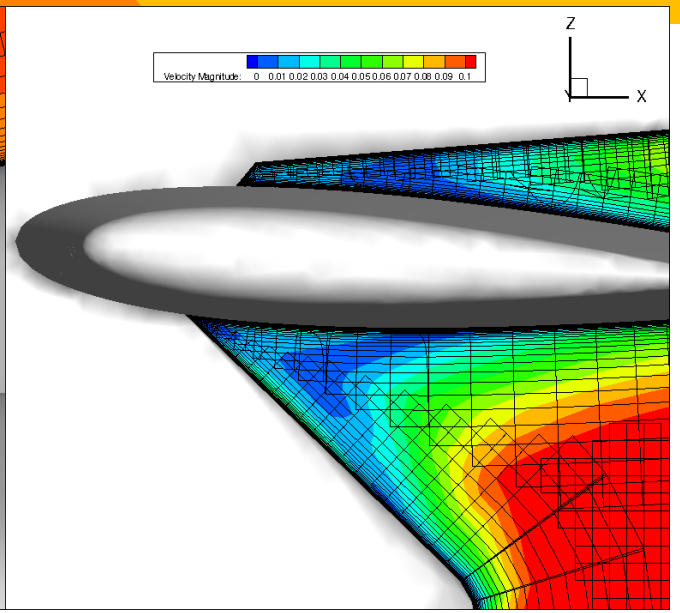
- **Advance Ratio = 0.23**
- **AOA = -4.31**

Time-accurate DES simulation

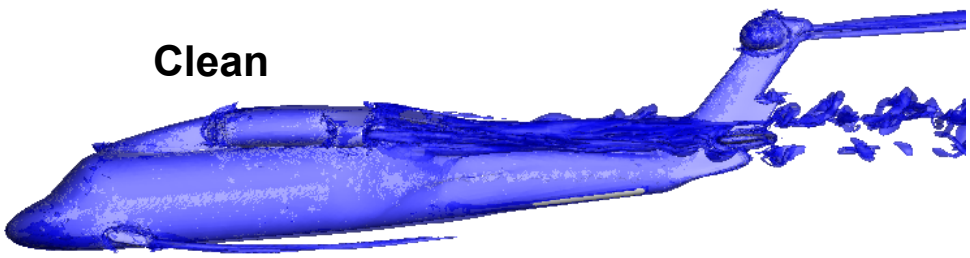
UH-60A Flow Solution: Clean Fuselage

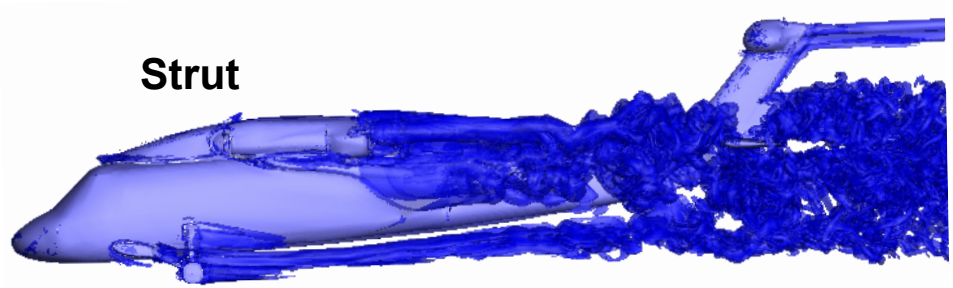


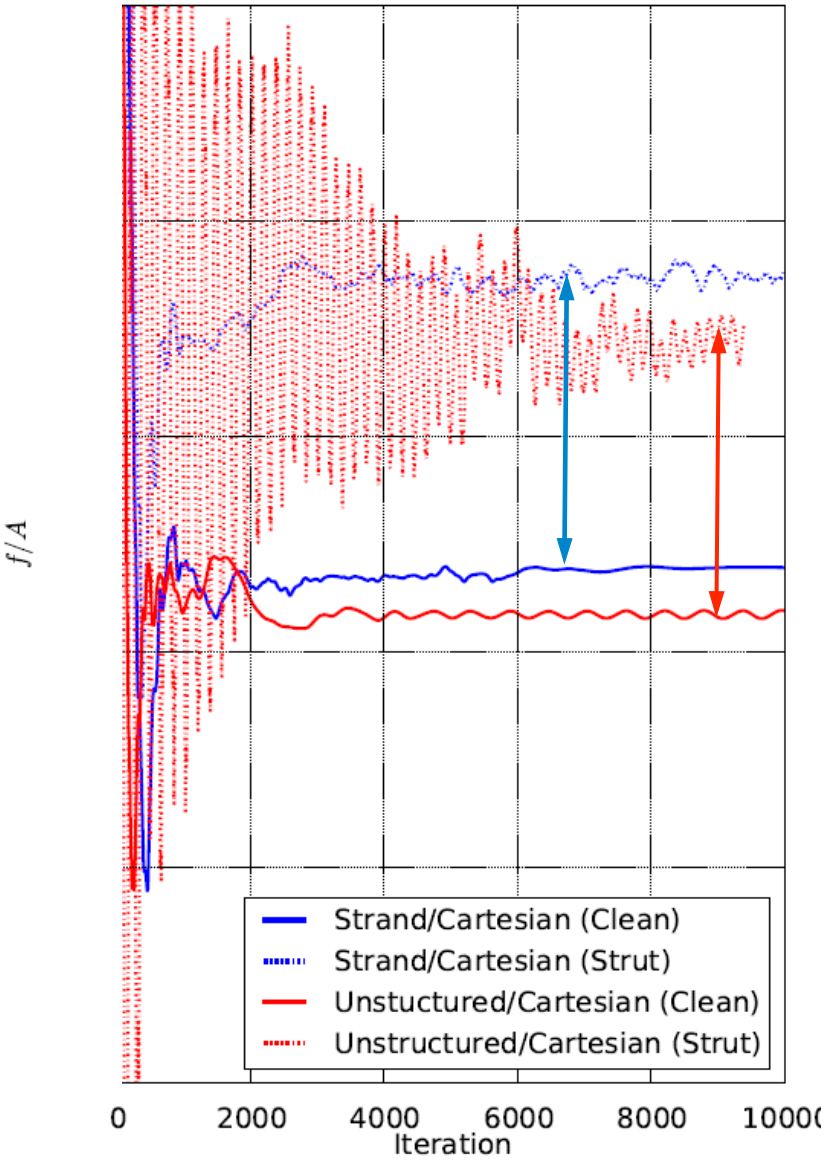




Drag Comparison: Clean vs. Strut, Unstructured vs. Strand



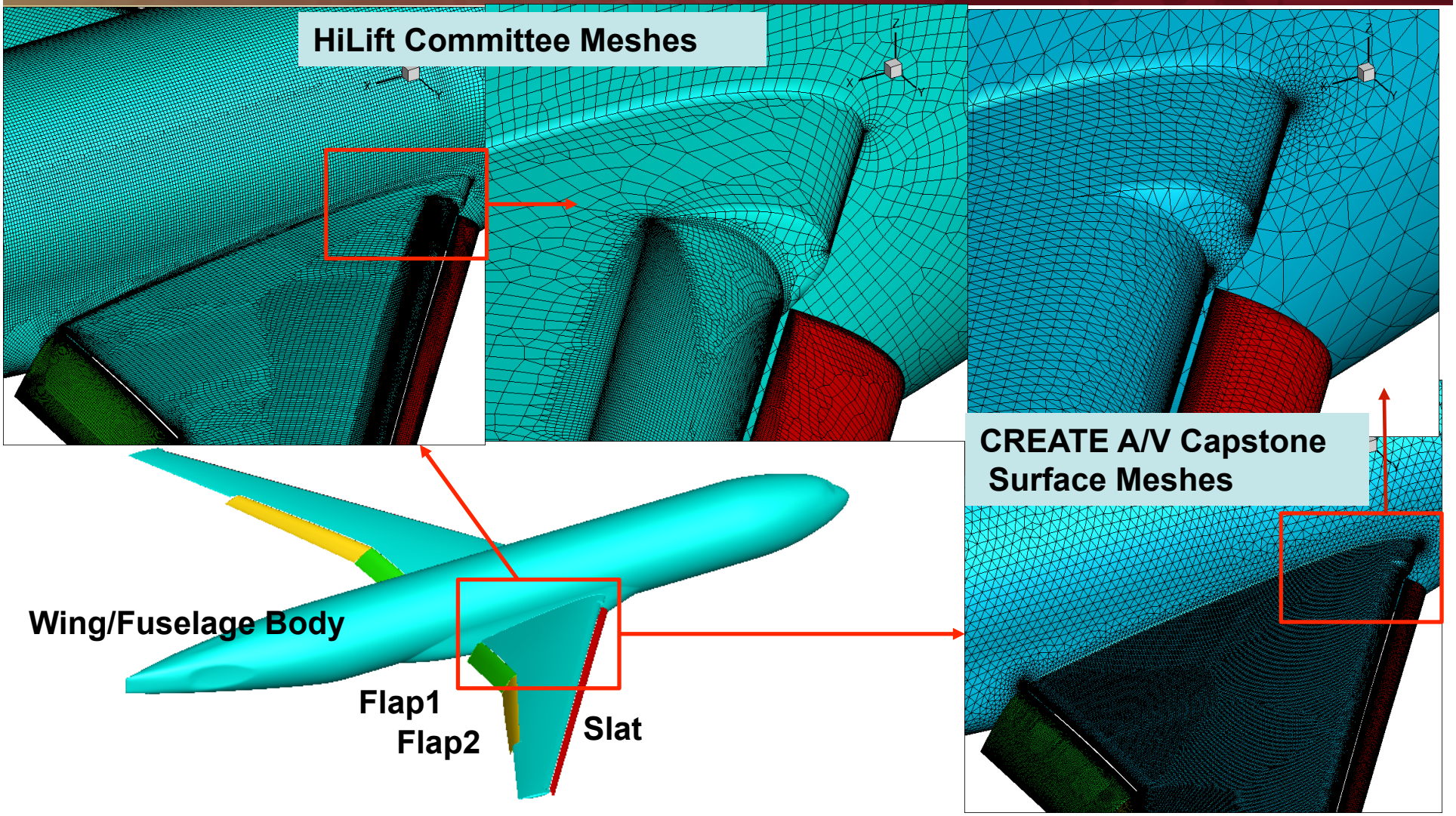




Drag prediction comparable between unstructured/Cartesian and Strand/ Cartesian

Available unstructured mesh is coarser than generated strand mesh

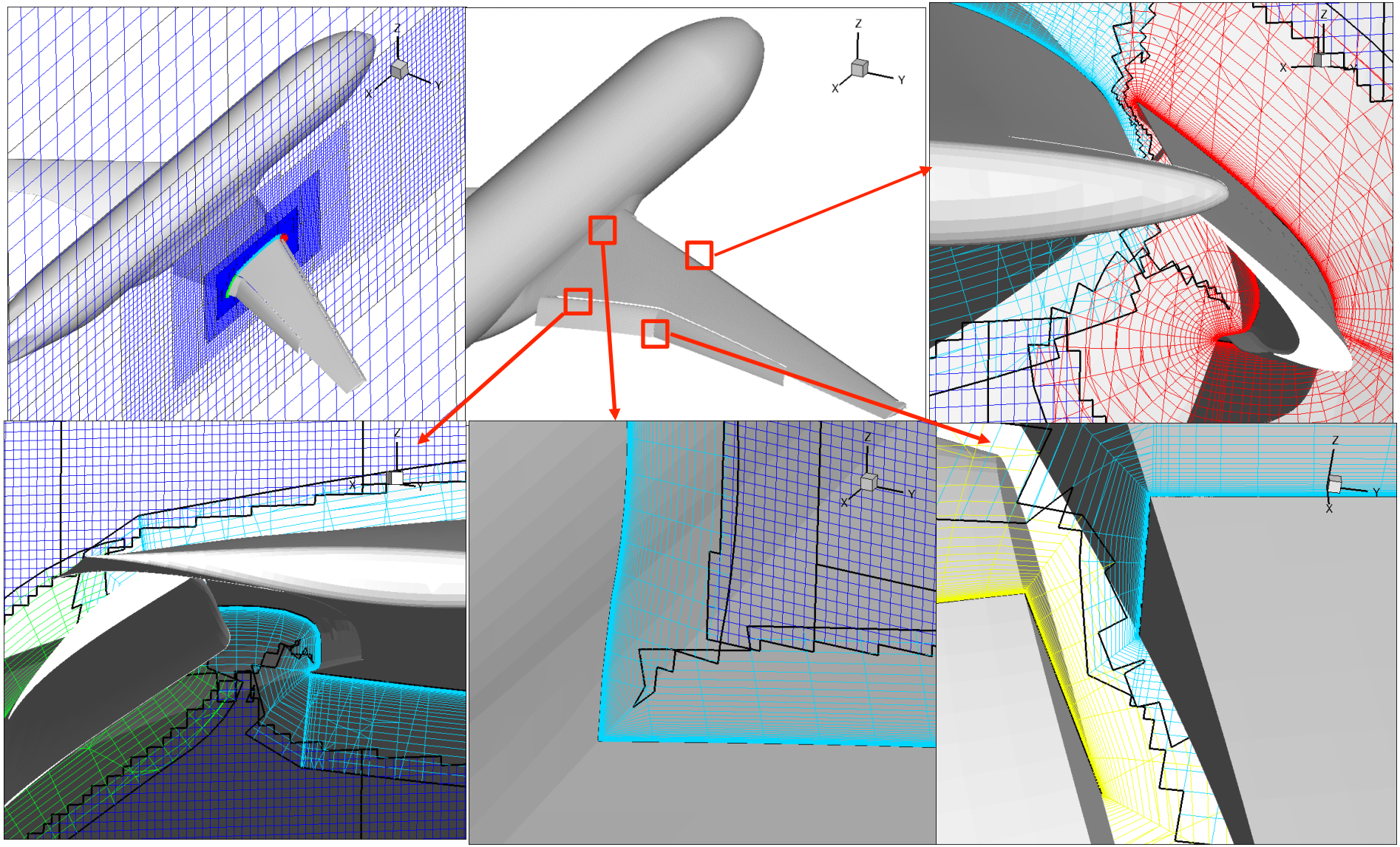
Case 4: HiLift Prediction Workshop (CRM)



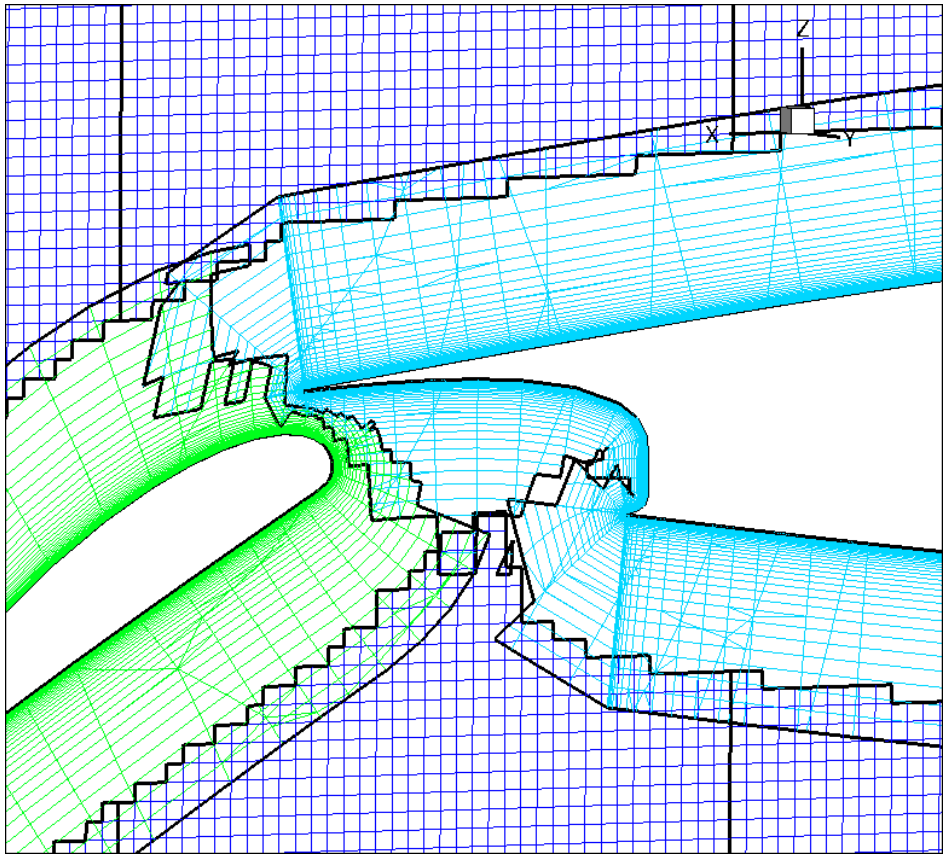
Goal: compare Strand/Cartesian calculations against Unstructured results

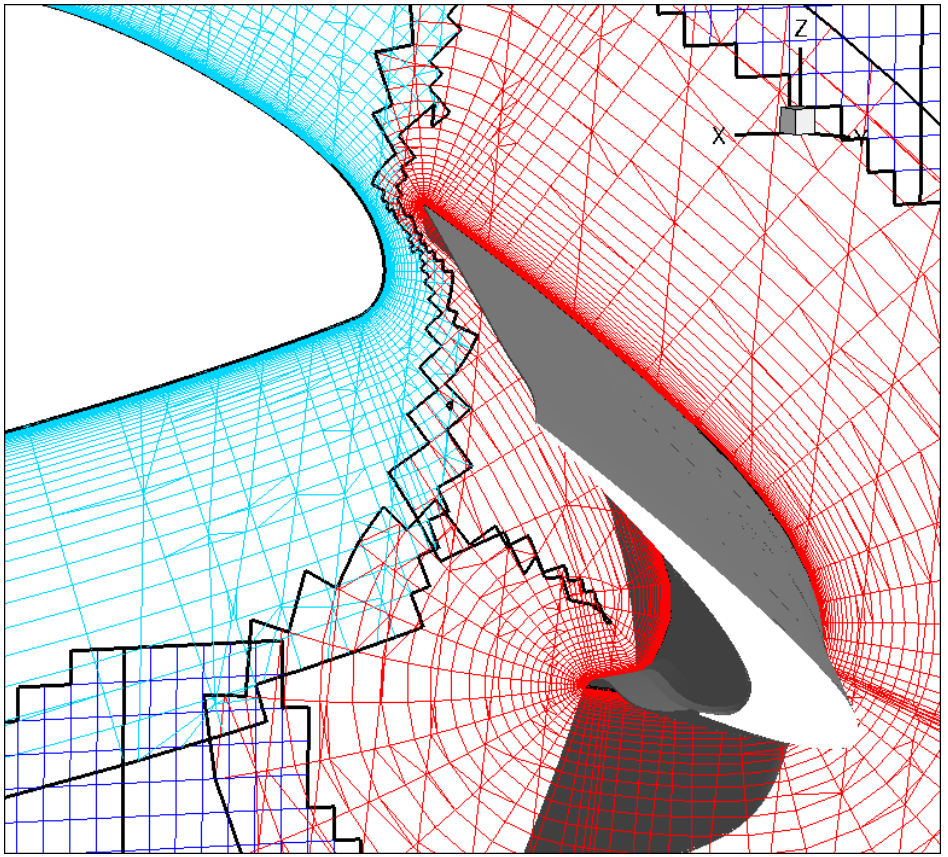
- **Surface meshes for Strand/Cart simulation generated using CREATE A/V Capstone.**
- **HiLift committee-provided volume meshes are used for unstructured calculation.**

Meshes and Domain Connectivity

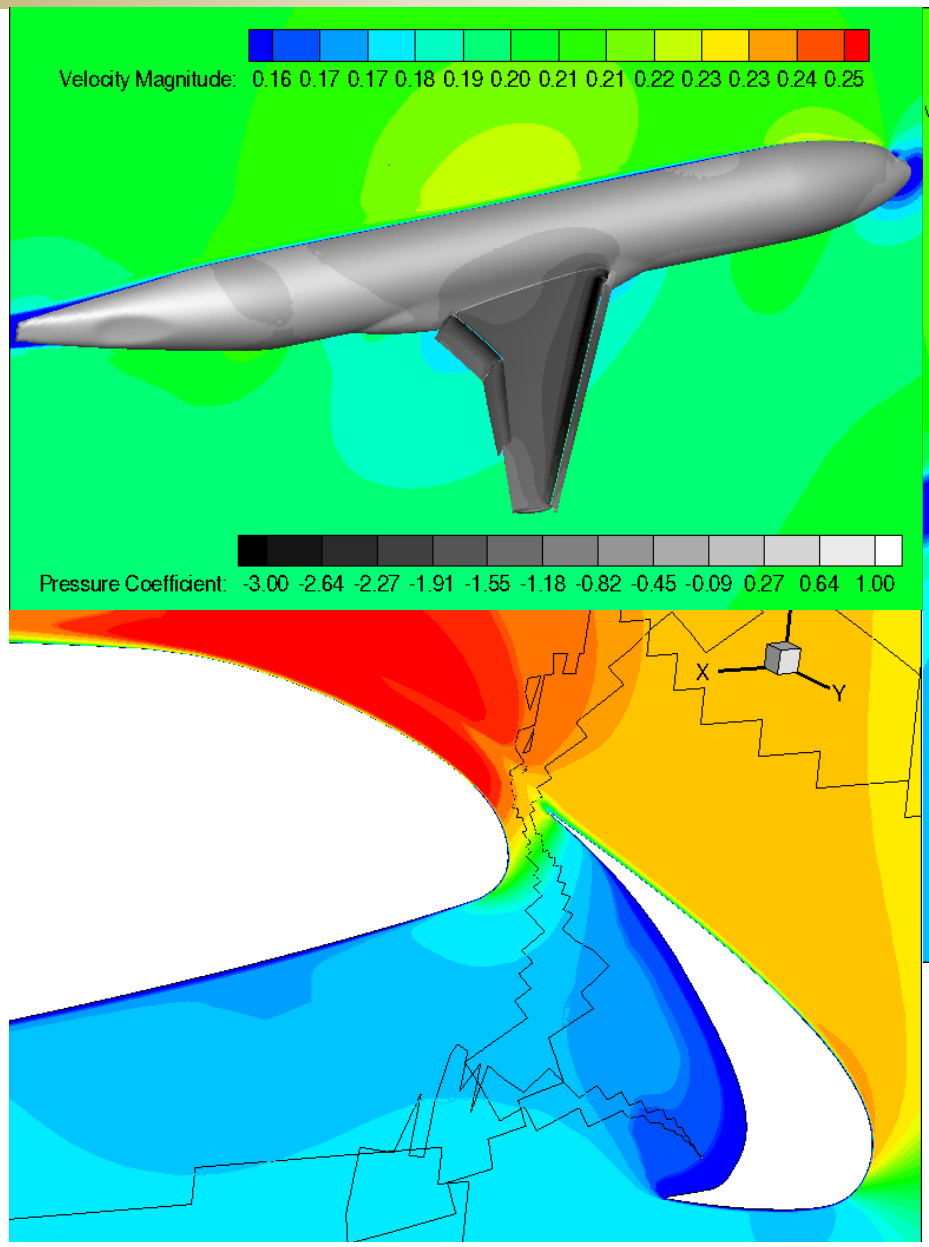


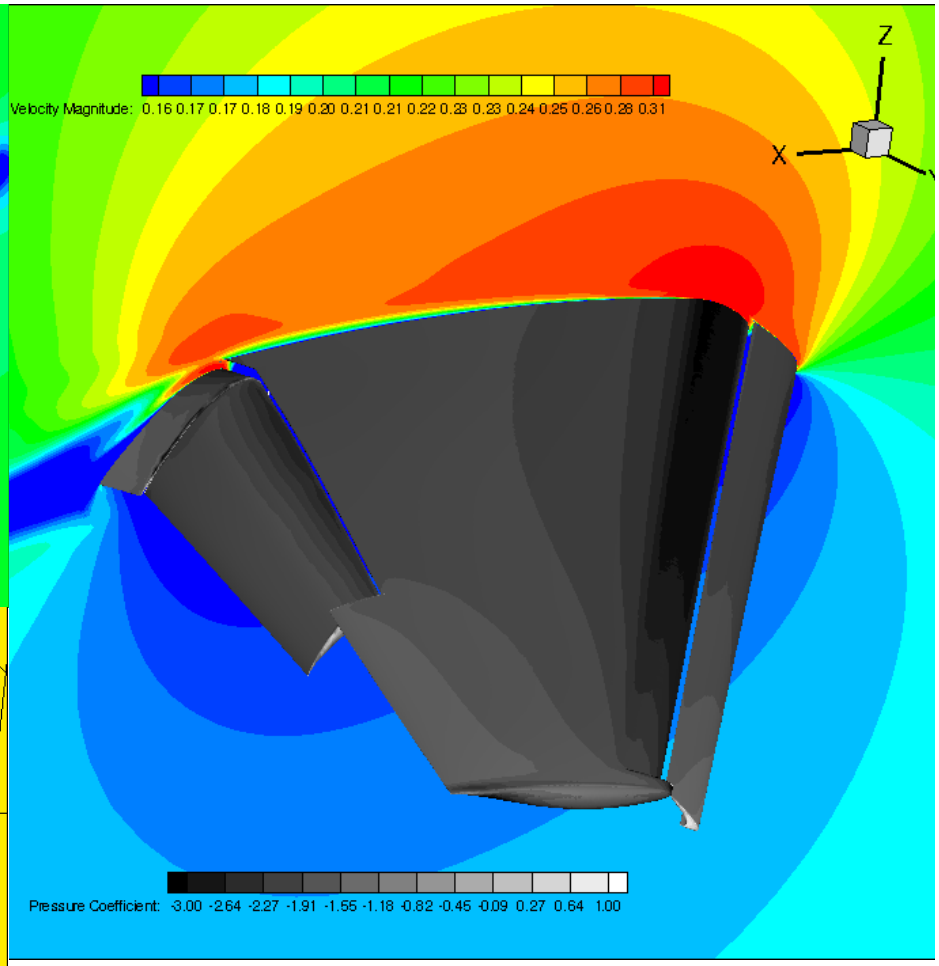
Meshes and Domain Connectivity



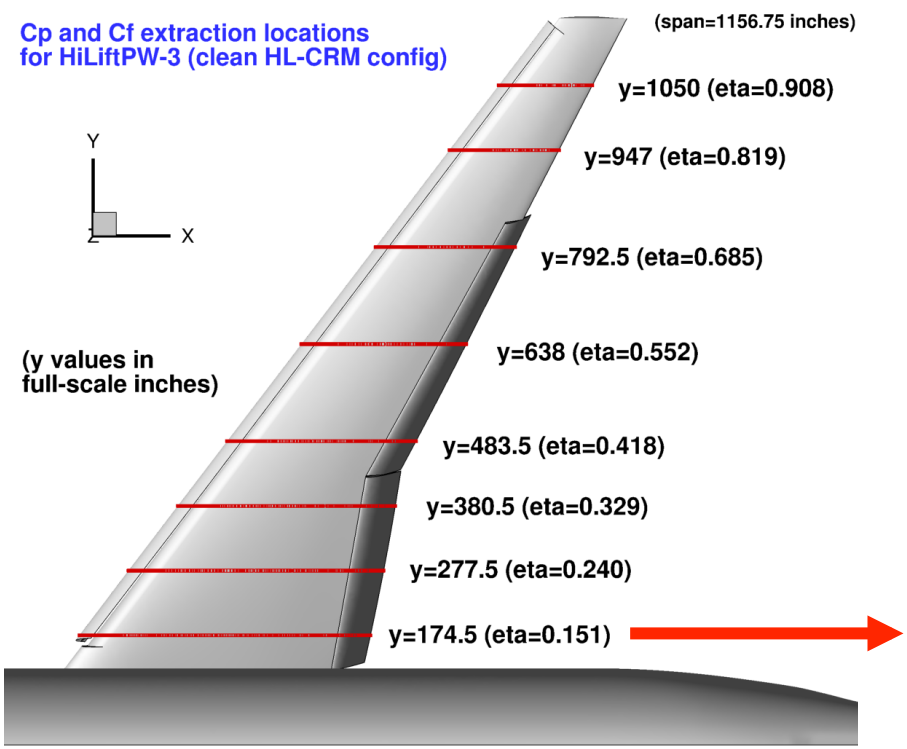


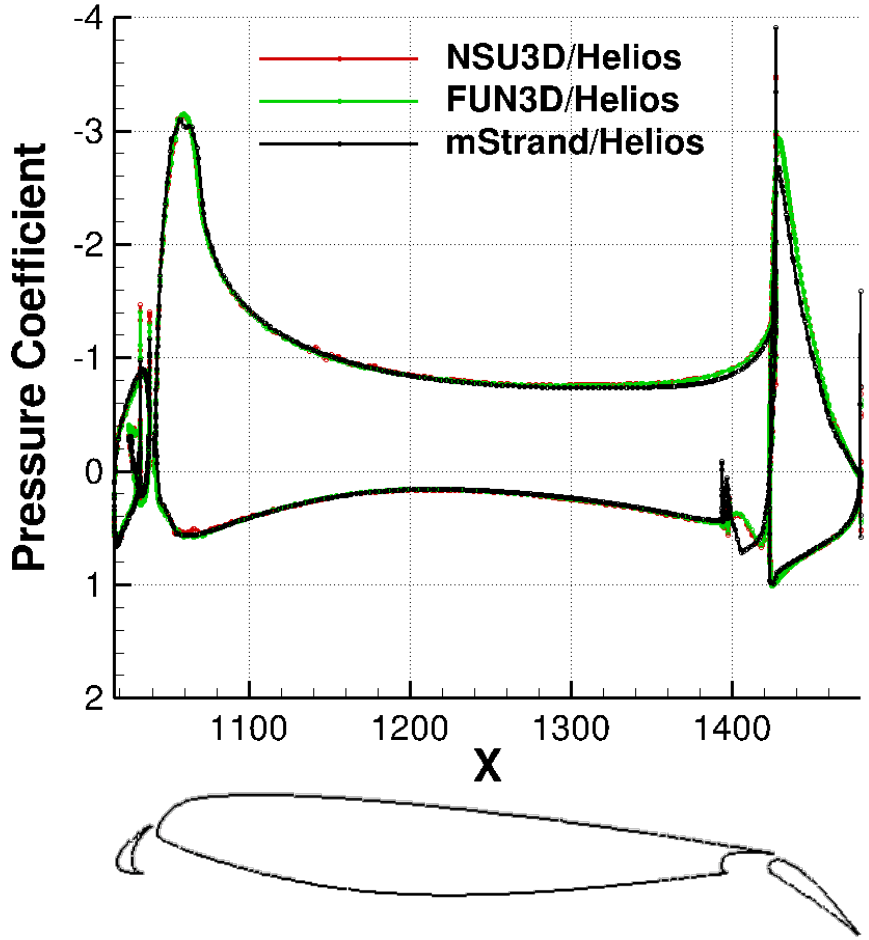
Flow Solution



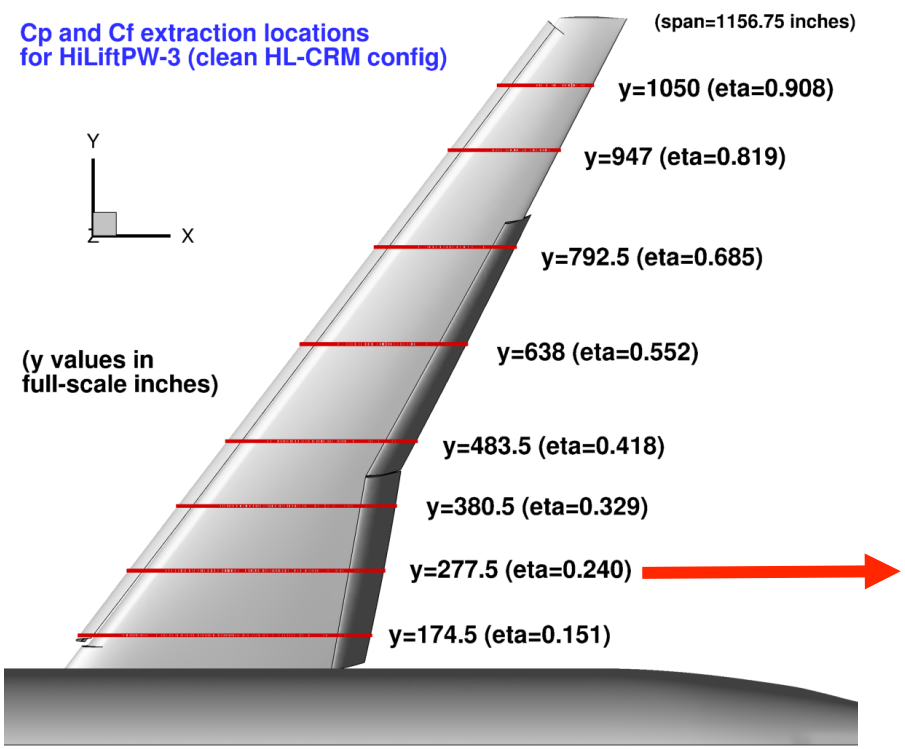


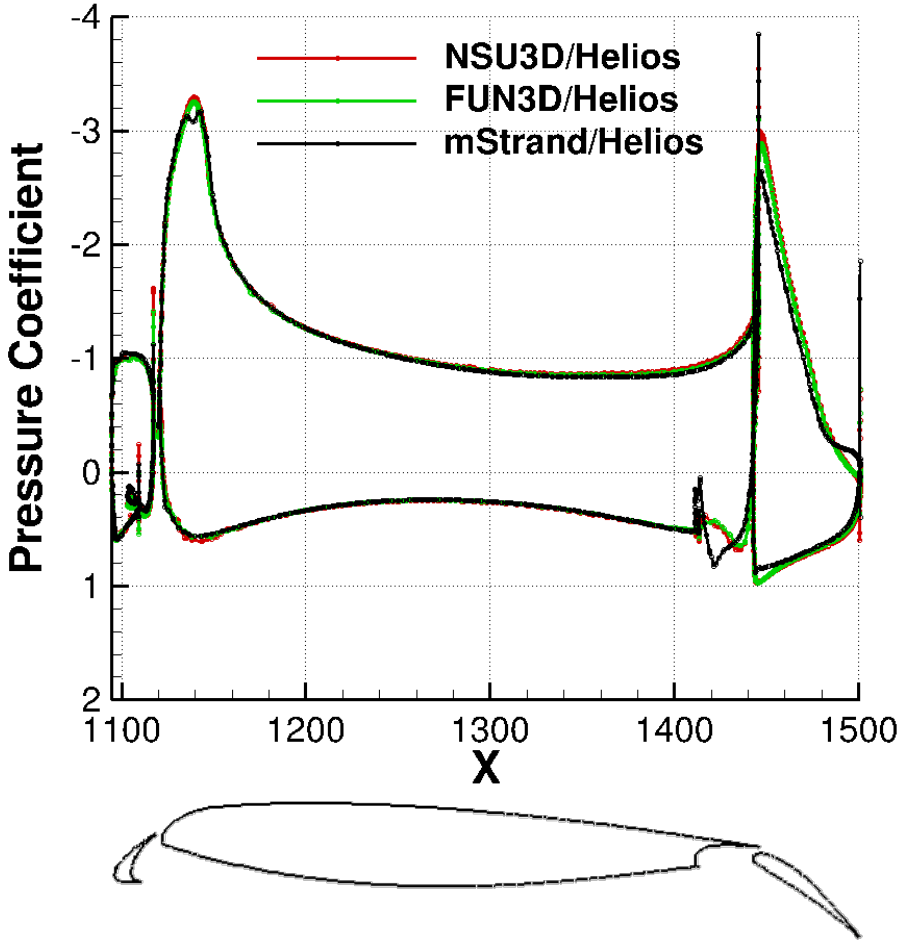
Pressure Coefficient Comparison



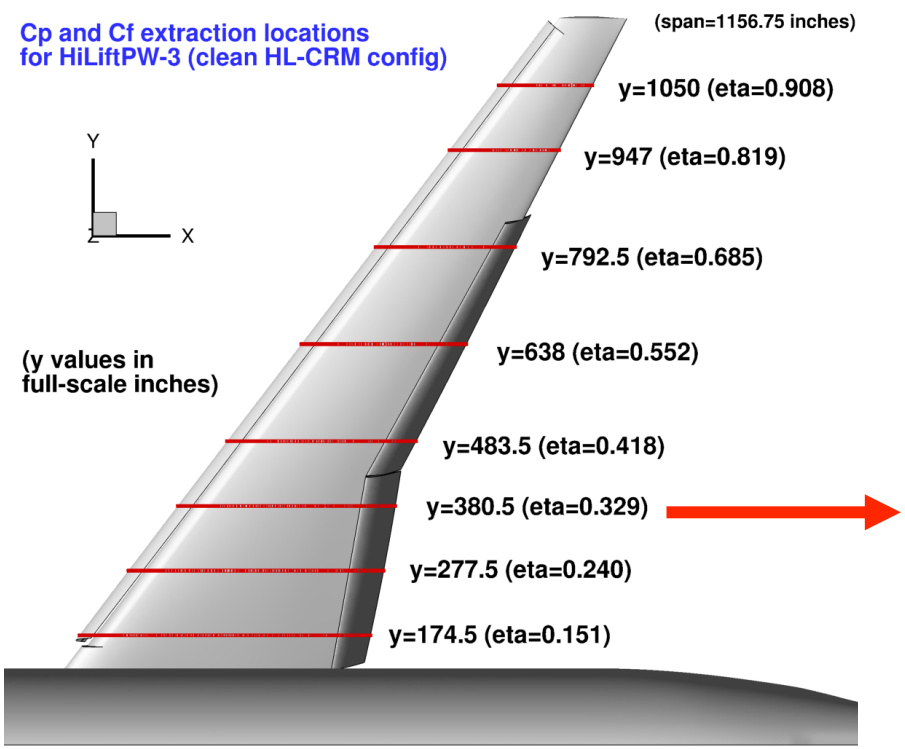


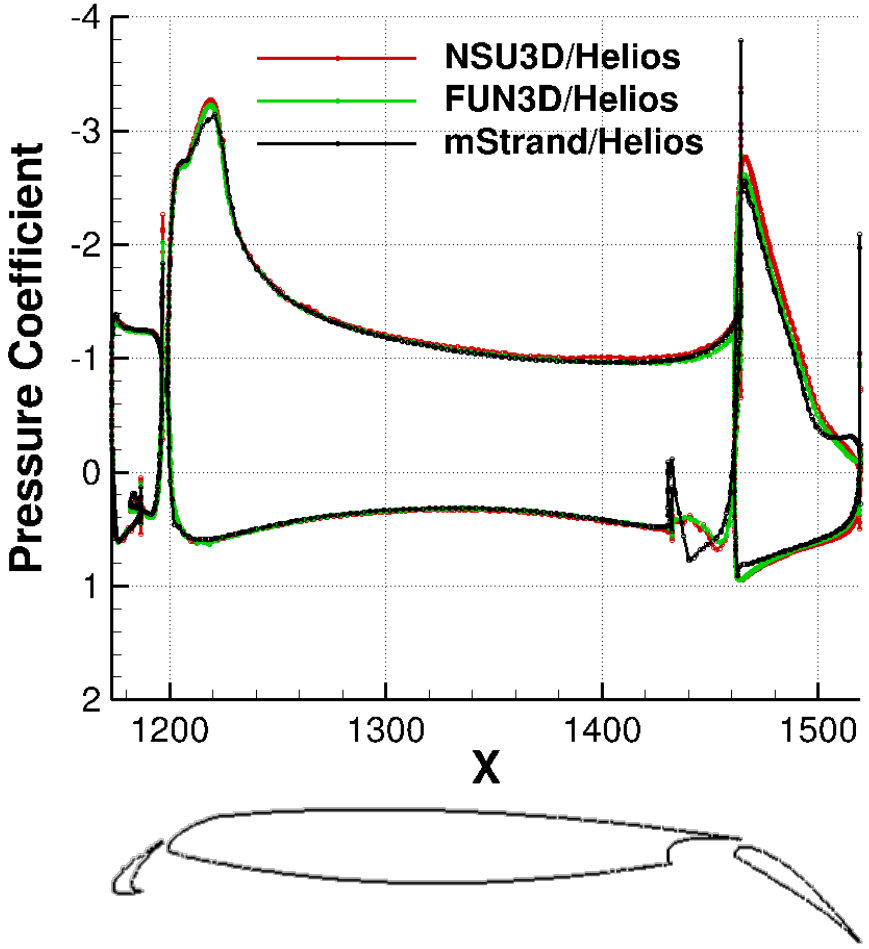
Pressure Coefficient Comparison



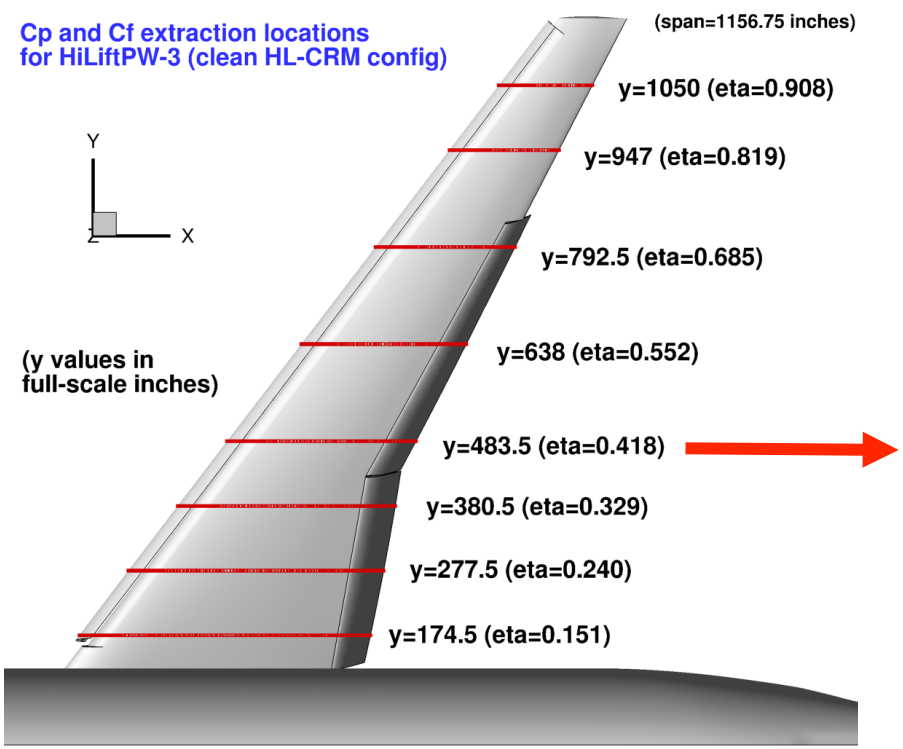


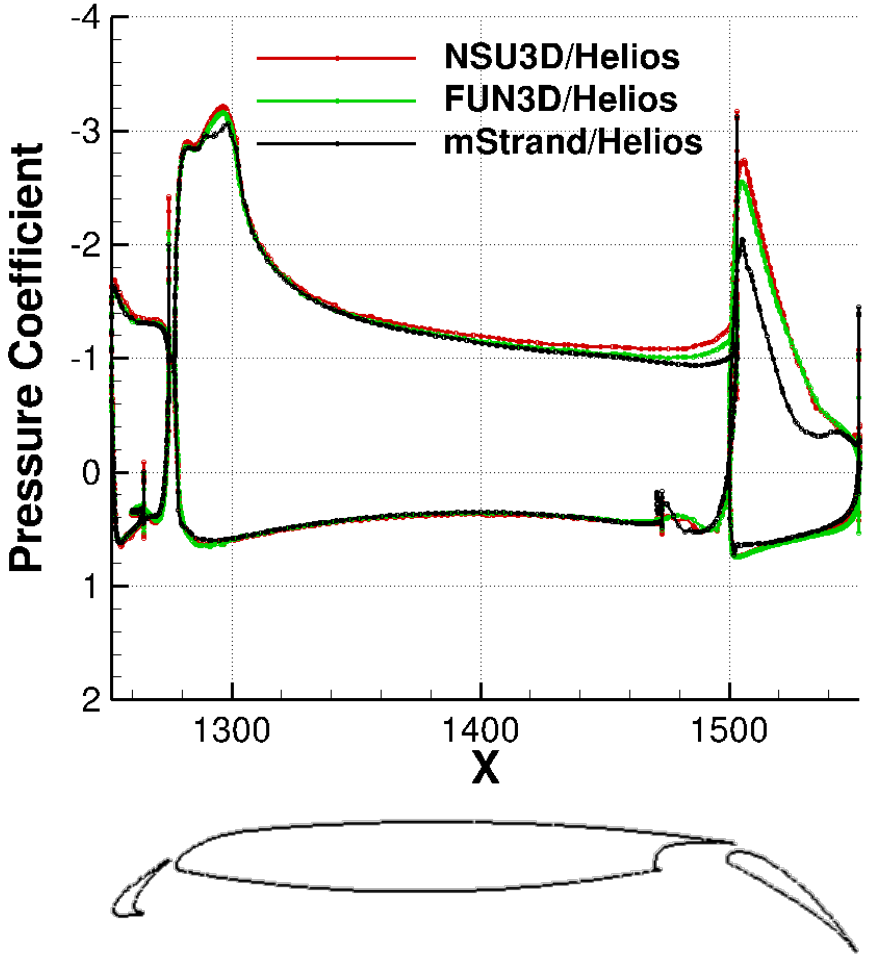
Pressure Coefficient Comparison



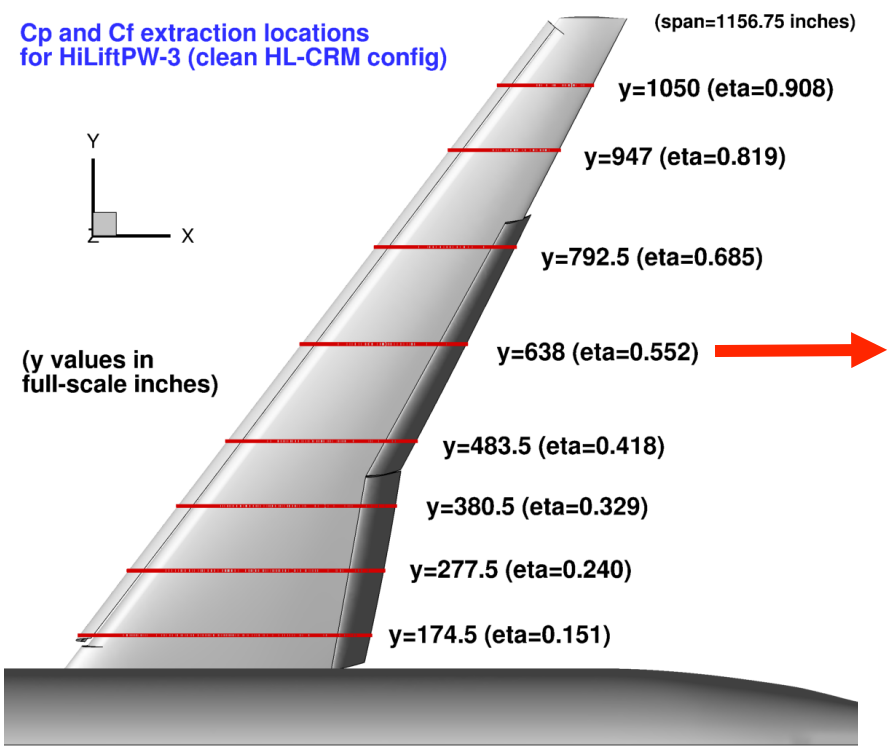


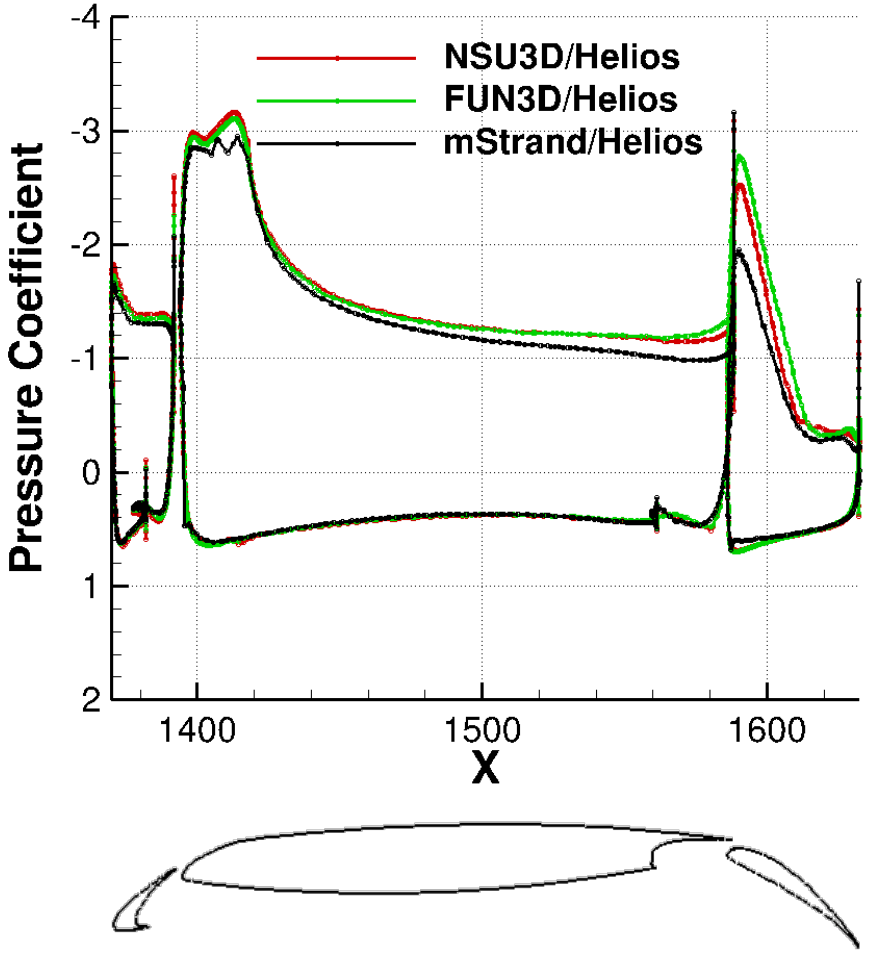
Pressure Coefficient Comparison



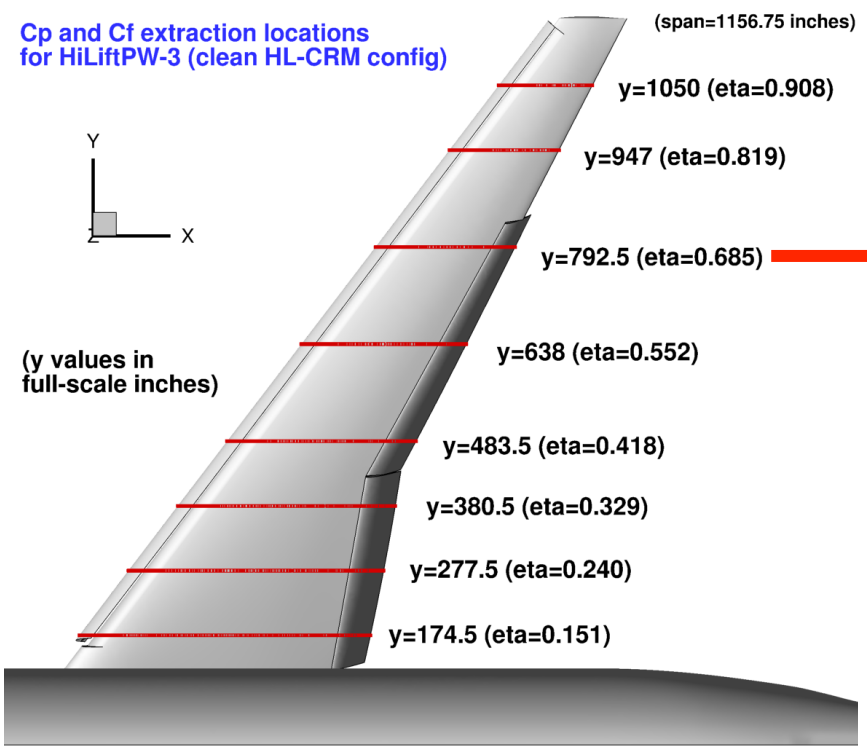


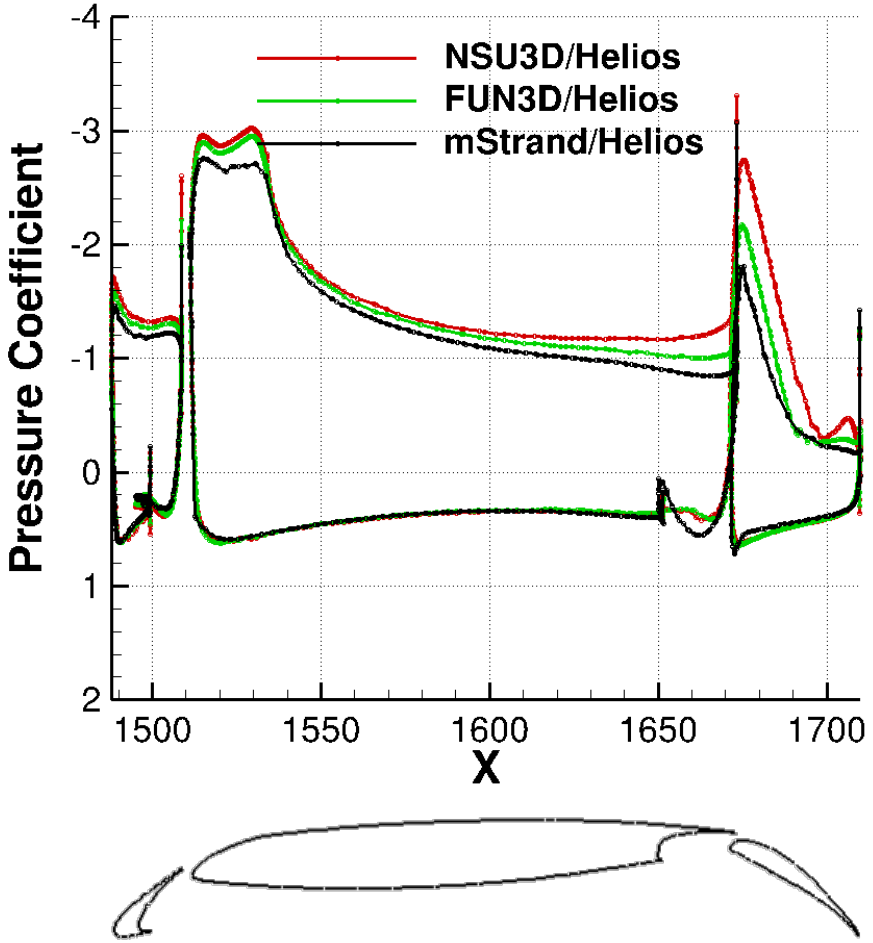
Pressure Coefficient Comparison



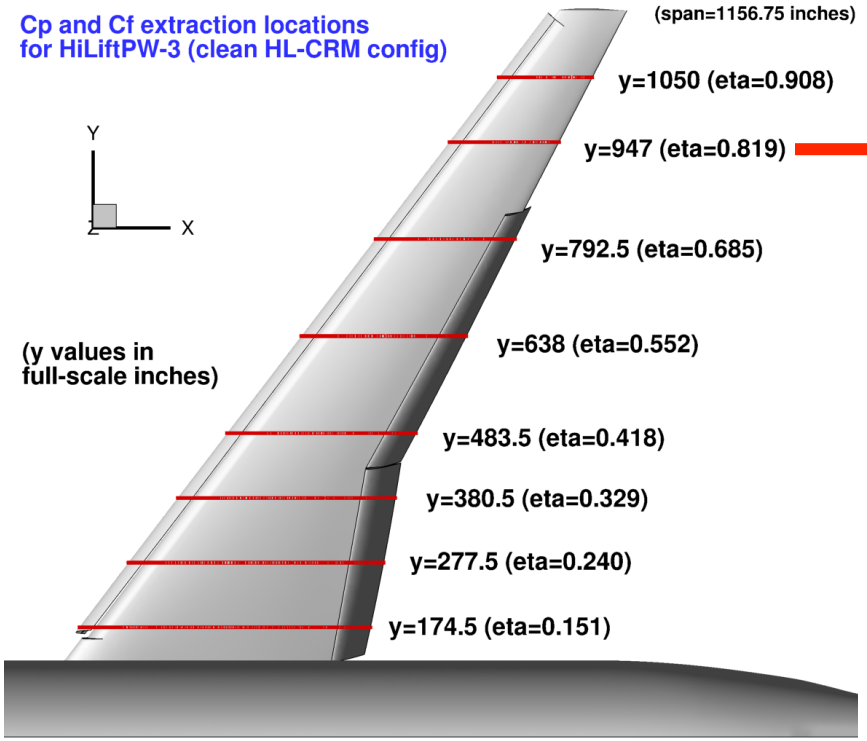


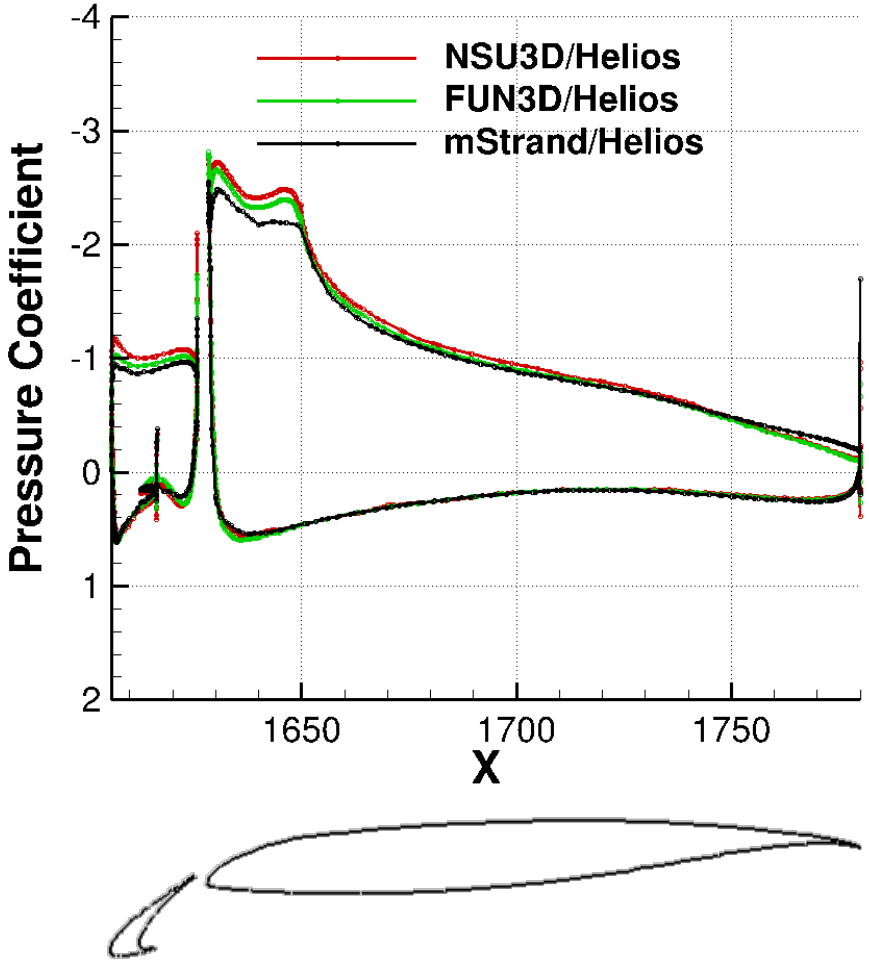
Pressure Coefficient Comparison



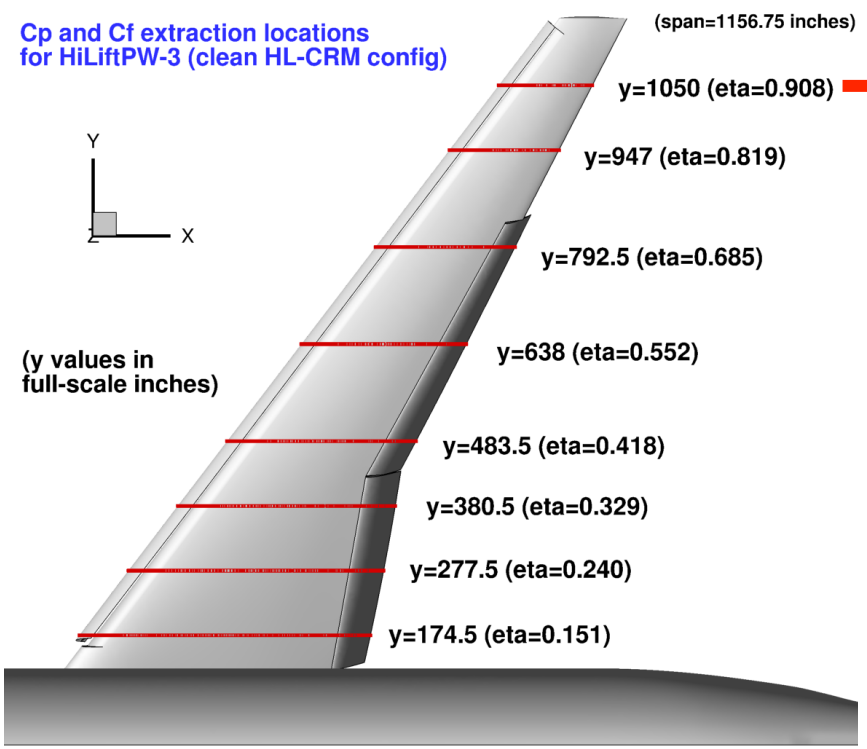


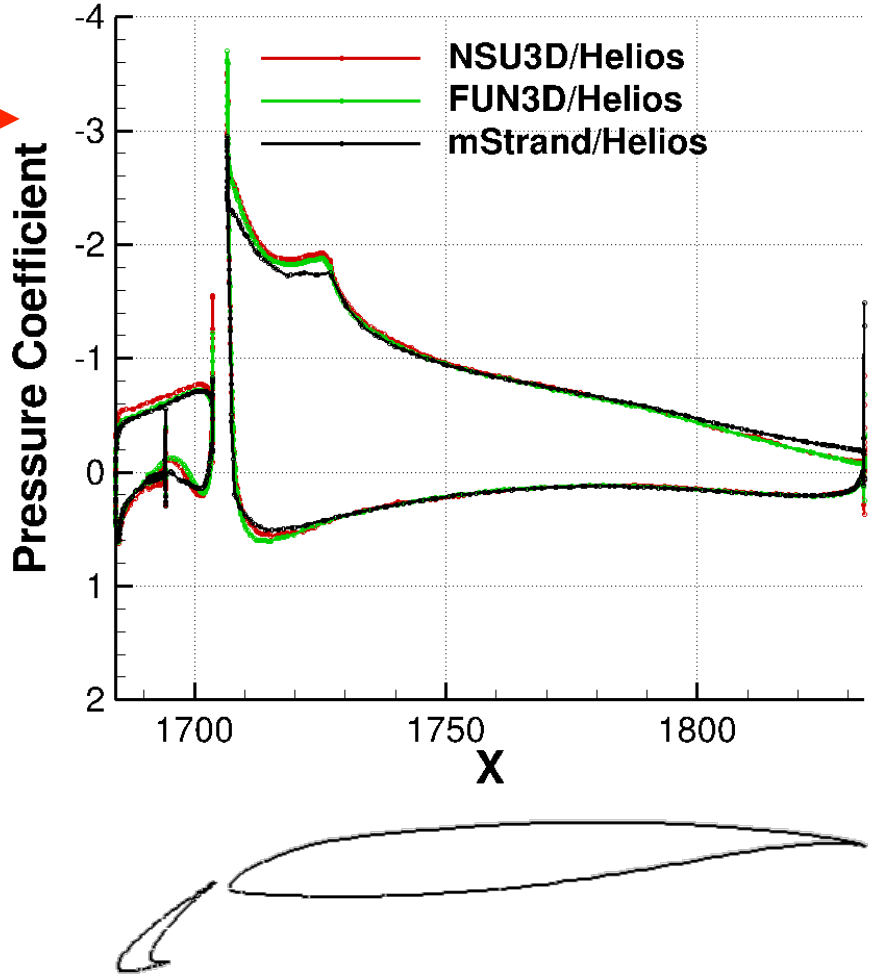
Pressure Coefficient Comparison





Pressure Coefficient Comparison





Case 5: UH-60A Rotor-Fuselage with Flexible Blades and Trim

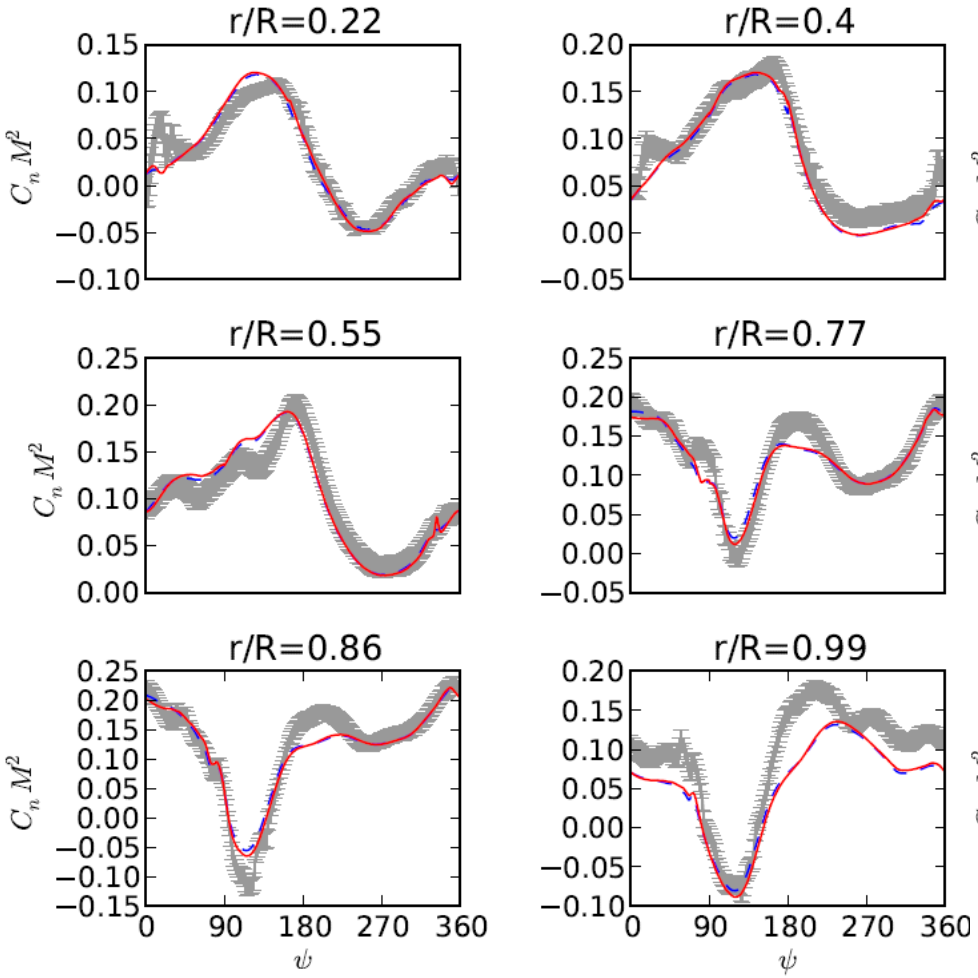




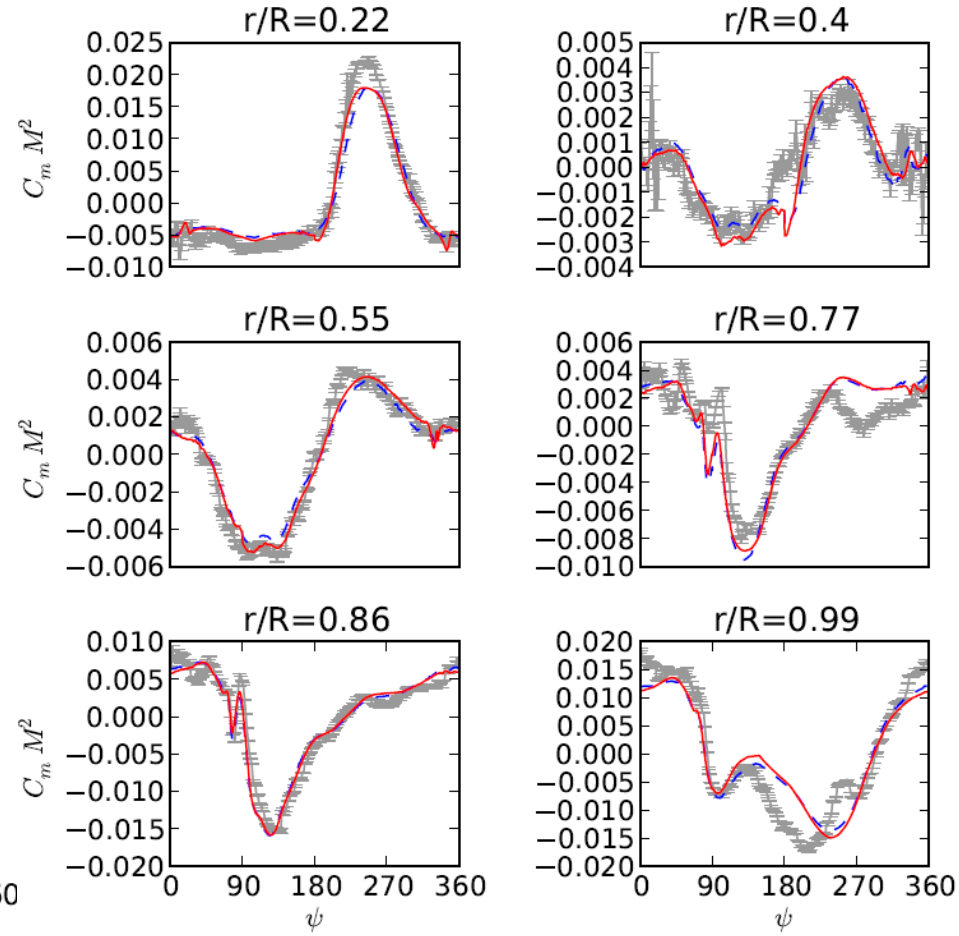
- Full aero-elastic test case
- All meshes are strand-based and auto-generated
- Coupled to structural dynamics (CSD) and trim
- Typical flight conditions and configurations used by industry designers
- High speed forward flight
- All options used
  Moving/deforming meshes
- Automatic intra-mesh and inter-mesh holecutting
- Off-body solution based AMR

Comparison with Baseline Unstructured/Cartesian Predictions

Sectional Normal Force



Sectional Pitching Moment

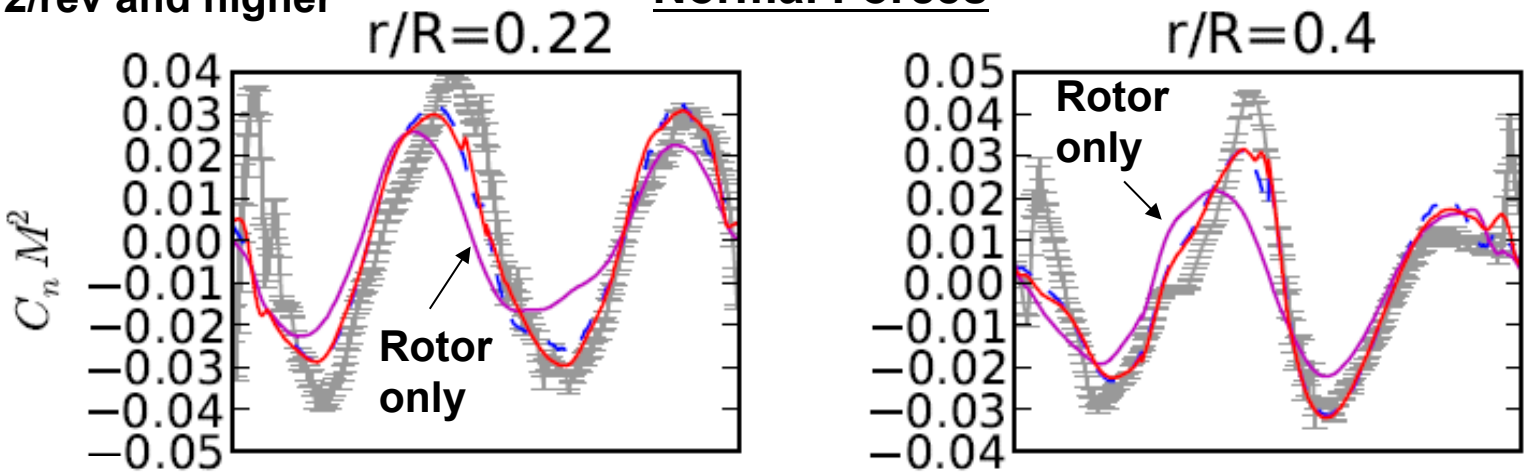


— Flt 8534
- - Unstructured/Cartesian
— Strand/Cartesian

Effect of Fuselage

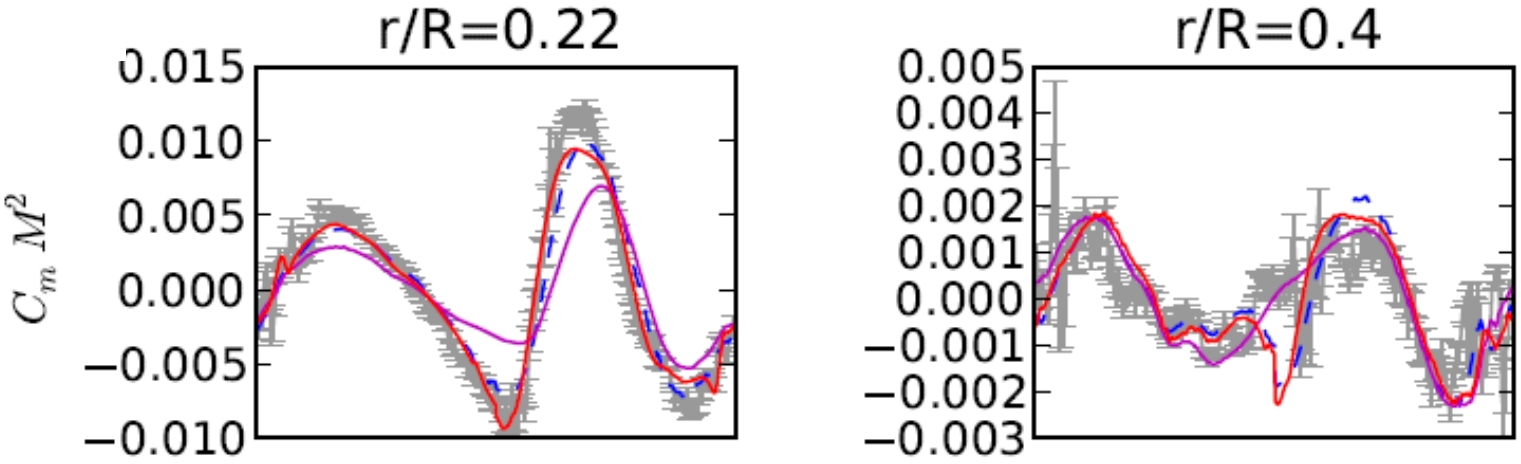
Vibratory loads 2/rev and higher

Normal Forces



- Flt 8534
- Unstructured/Cartesian
- Rotor-only
- Strand/Cartesian

Pitching Moments



Concluding Remarks

- **Progress towards fully automated Strand meshing for complex geometries**
  - Closest Vertex on Isosurface of Distance I_L
  - Elastic Smoothing constrained on I_L
  - Strand collision problem solved via intra-mesh domain connectivity
- **Validated solution accuracy for both intersecting and non-intersecting meshes**
- **Demonstrated equivalent accuracy to unstructured methods using auto-generated volume meshes**
- **Extended simulations to realistic complex test cases**

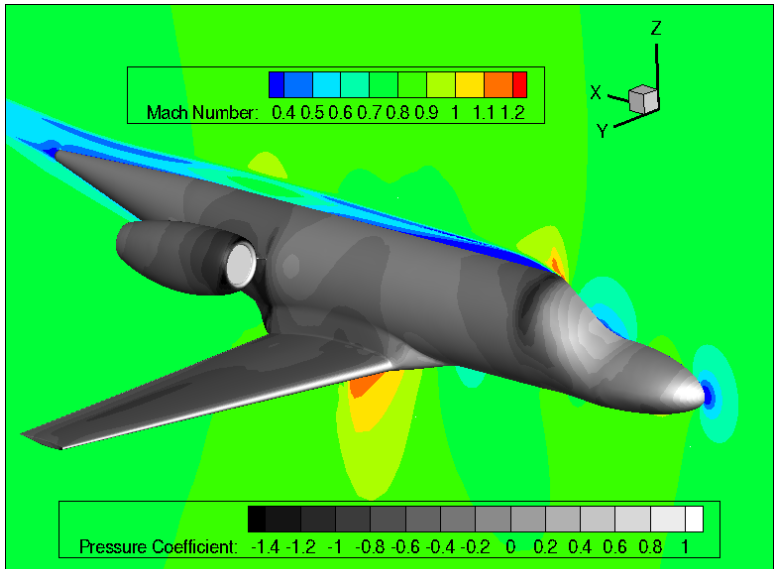


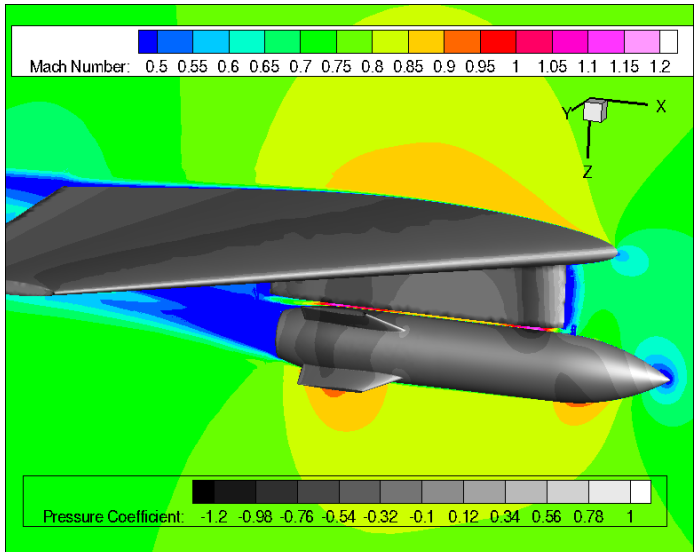
<u>Challenges</u>:

- **Good surface meshes are required for good strand mesh. Have to tie surface and volume meshing to CAD**
- **Need to understand/resolve differences between intersecting and non-intersecting meshes**

Future Work

- Stress-test the method for many complex geometry problems
- Fully automate decision logic for
  - inner/outer layer thickness
  - number of levels / level thickness
- Validation and Verification for other known cases



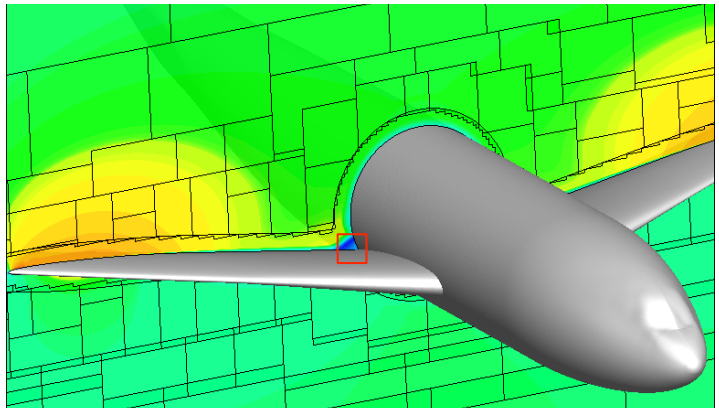


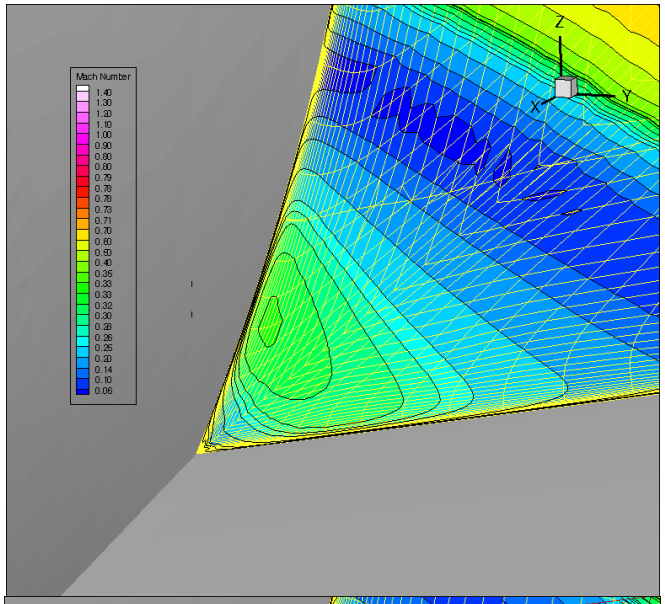
Acknowledgements

Support from the HPCMP CREATE A/V program and CREATE MG Capstone team

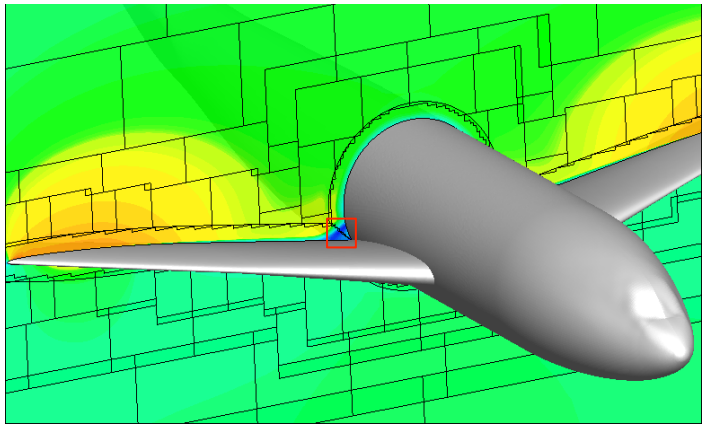
BACKUP SLIDES

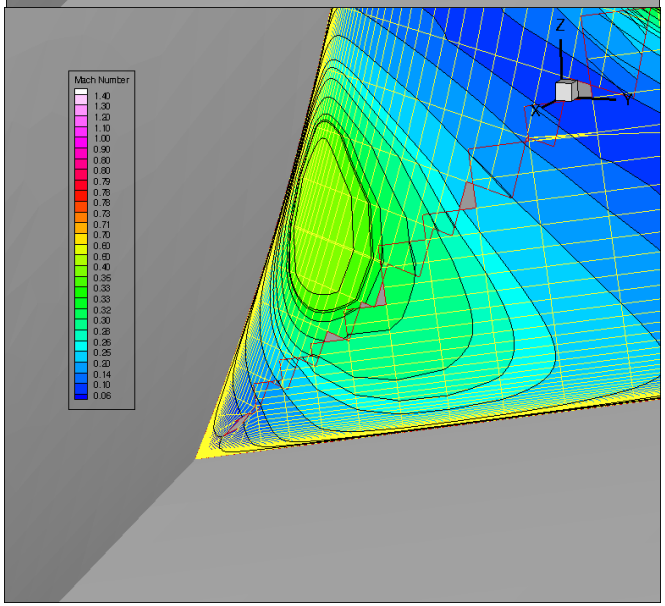
Non-intersecting





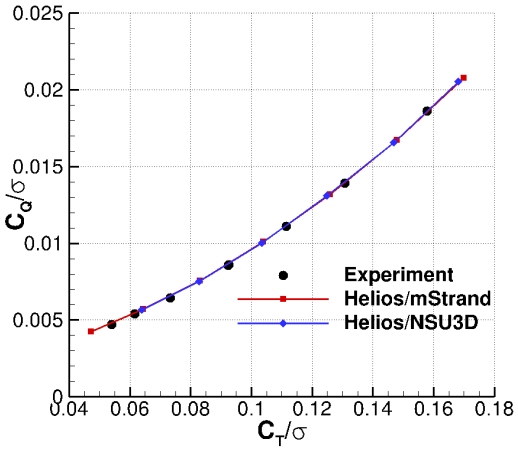
Intersecting

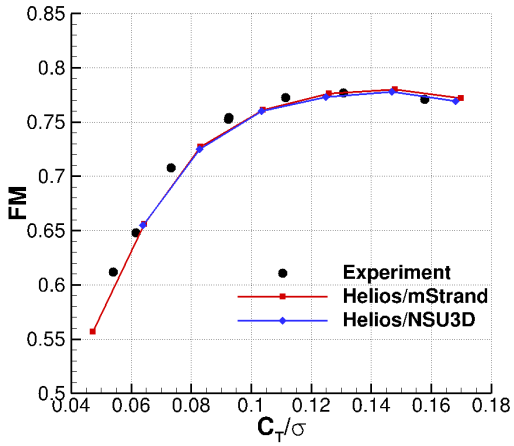


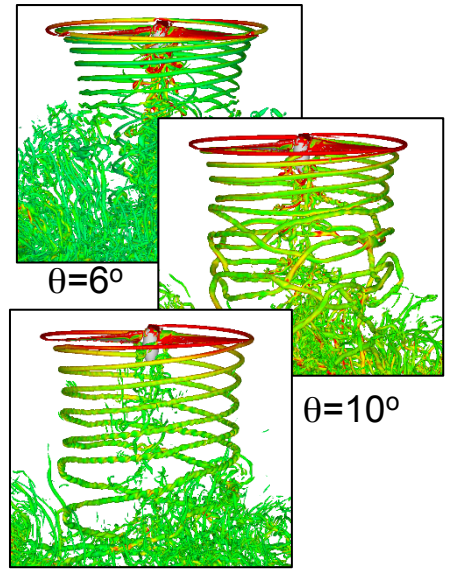


mStrand Solver

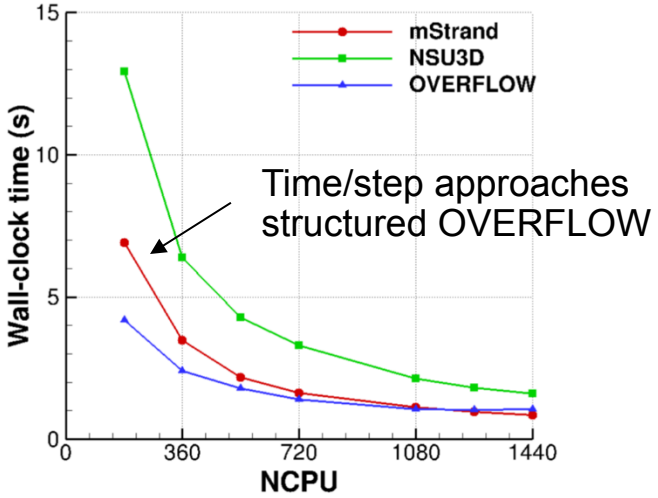
- **Accuracy commensurate with best solutions we obtain with NSU3D**

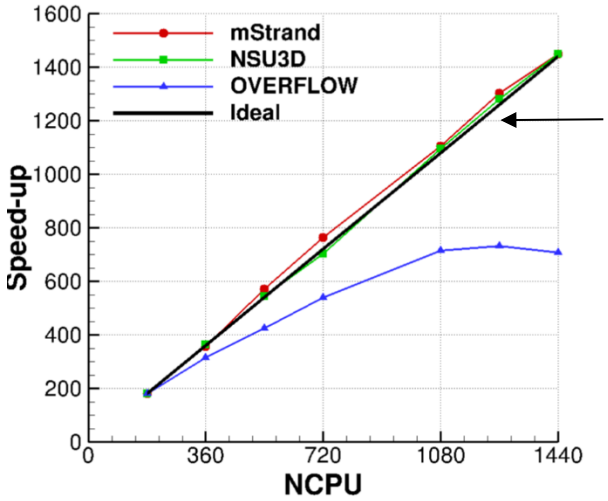






- **Good computational performance**





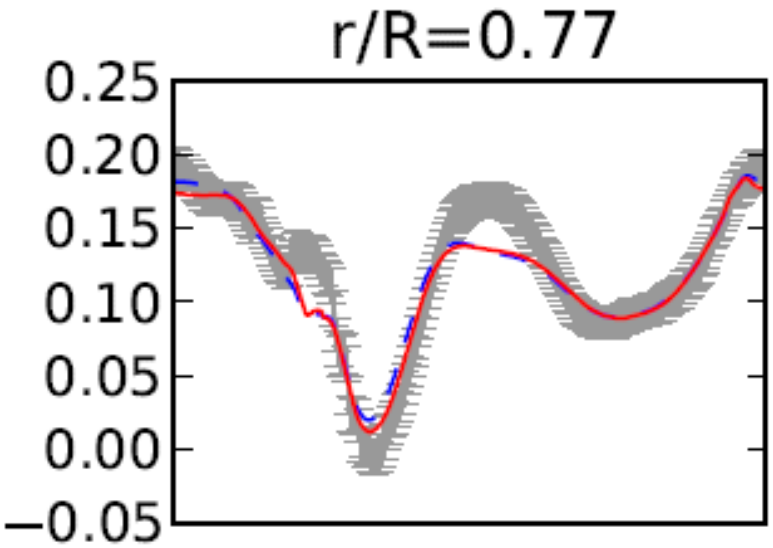
Scalability on par with unstructured NSU3D

More details

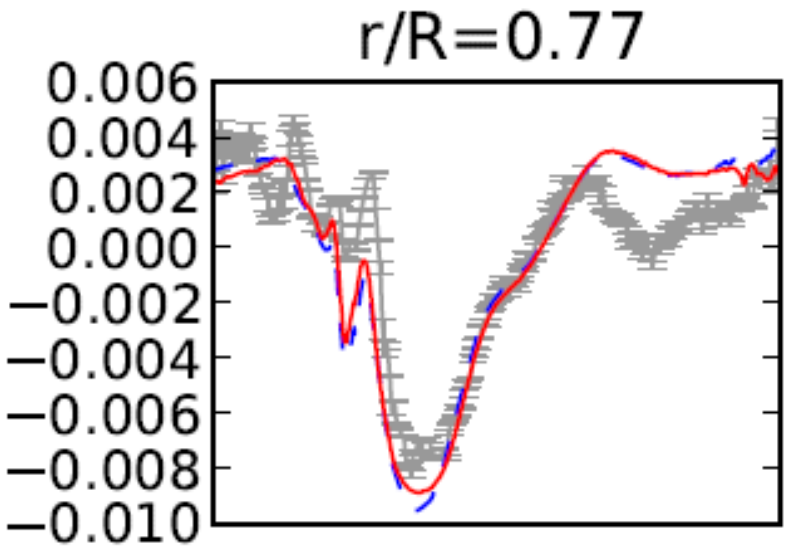
Lakshminarayan et al
AIAA-2016-1581

Comparison with Baseline Unstructured/Cartesian Predictions

Sectional Normal Force



Sectional Pitching Moment



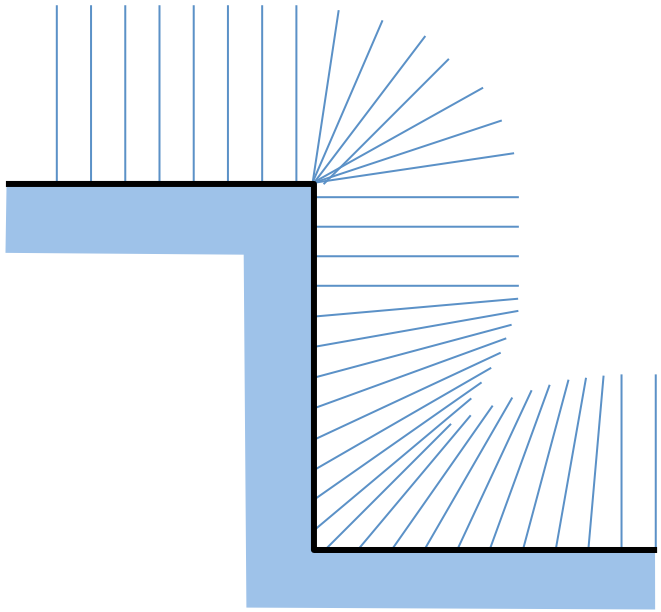
Flt 8534

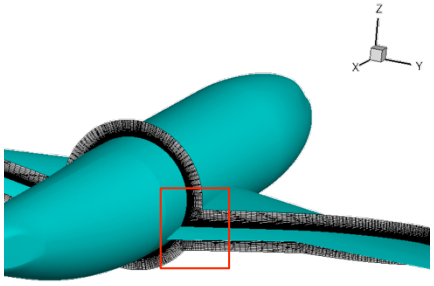
Unstructured/Cartesian

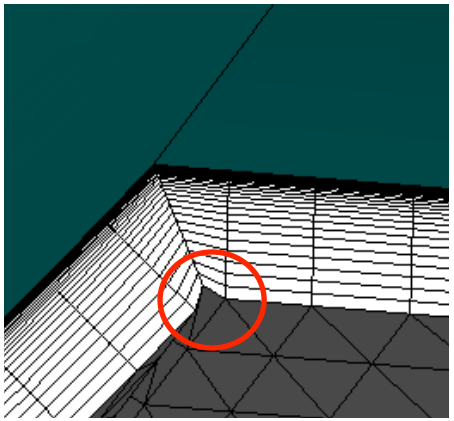
Strand/Cartesian

Motivation: Need to Amend MOSS Approach

- Minimize top surface area at problematic locations keeping stand length constant = L (details in Haimes, Overset Grid Symposium, 2014)

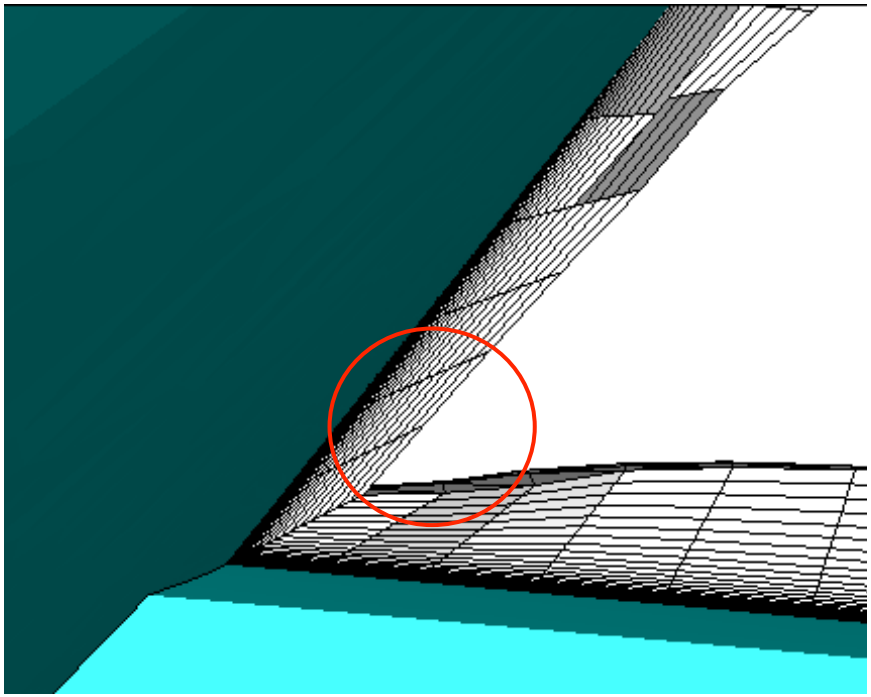




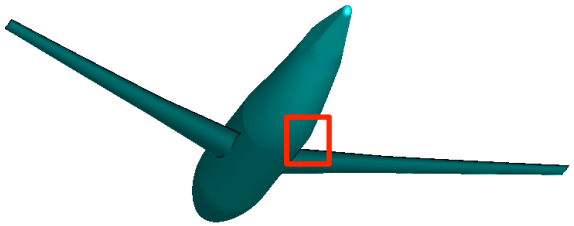


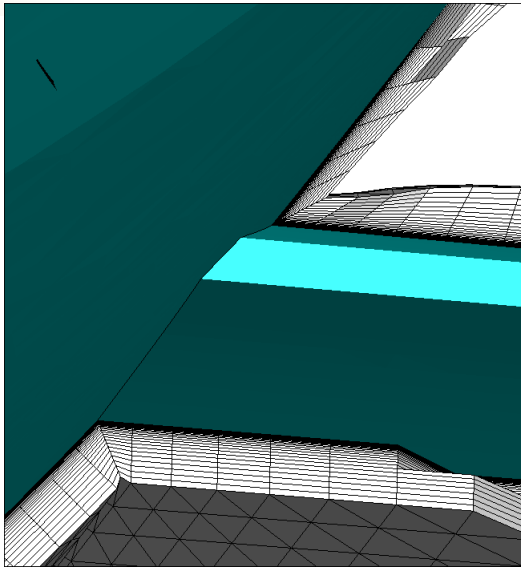
Challenges:

- Concave regions → deep creases because of strand length constraint
- Difficulty creating next strand level

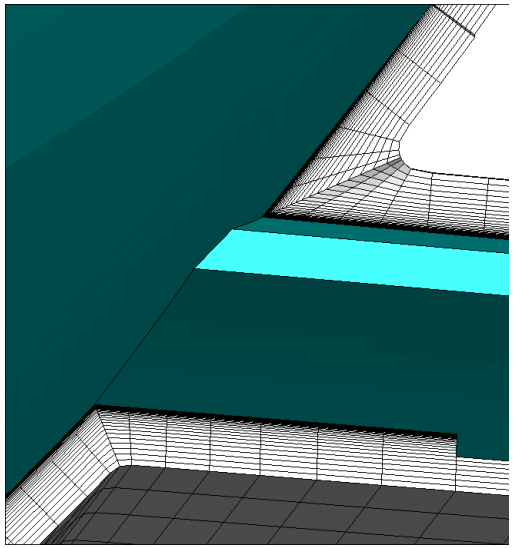


Meshing Improvement With CLOVIS Method

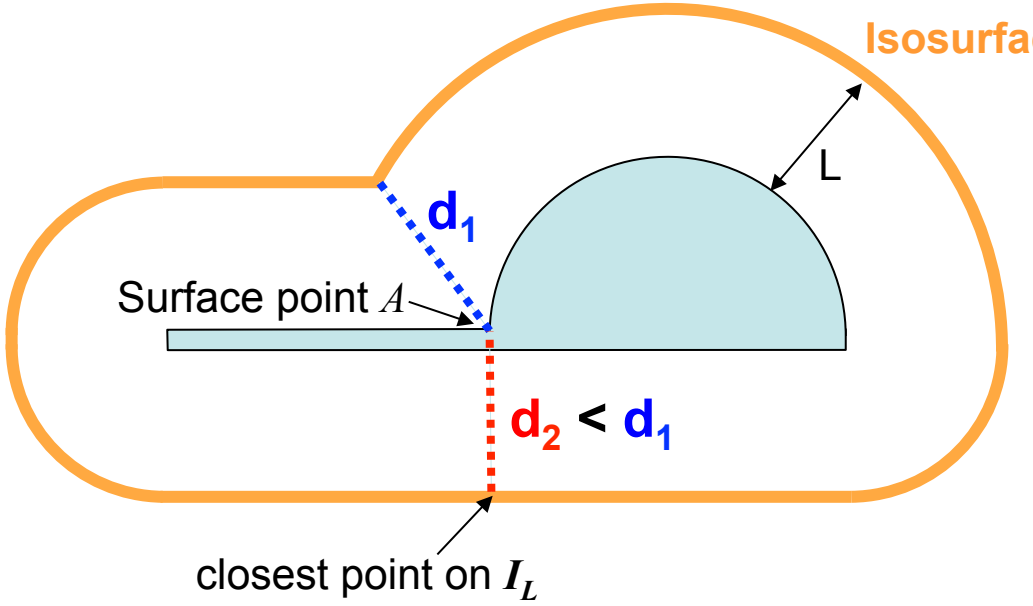




MOSS



CLOVIS



Increased concavity on the outer surface with constant length strands

CLOVIS produces significant improvement

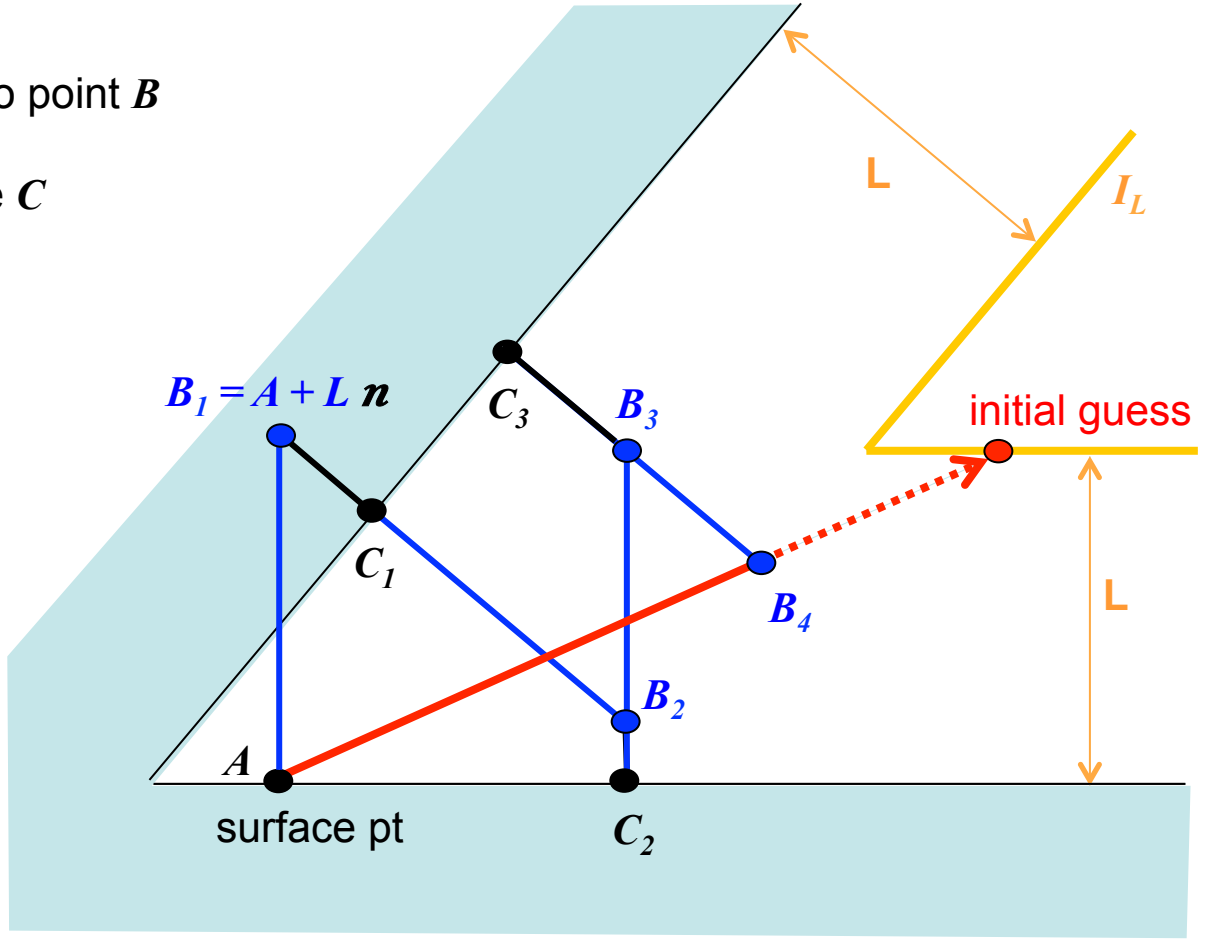
How to solve CLOVIS?

Finding the Closest point on I_L

Problem: For each surface point A, find P, closest point on I_L
(isosurface of distance field at L),without actually constructing the isosurface

Find initial guess:

- Follow local normal n for L to point B
- Find closest point on surface C
- Repeat until convergence
- Flush point B to isosurface
  (search along vector CB
  until distance to S = L)

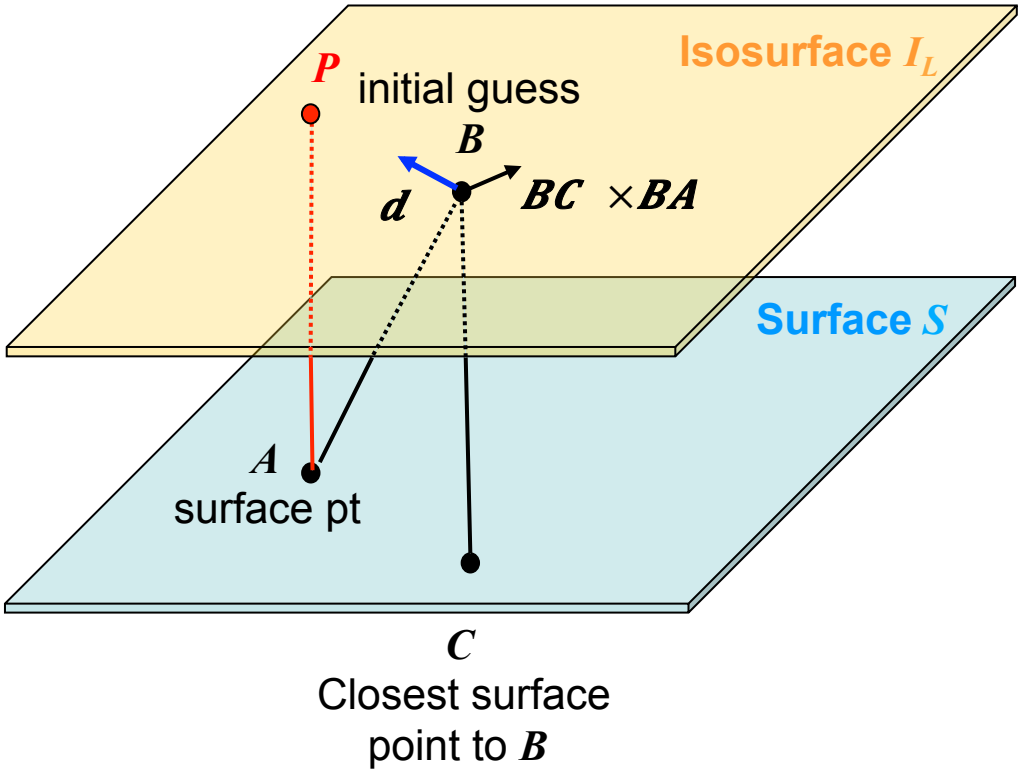


Finding the Closest point on I_L

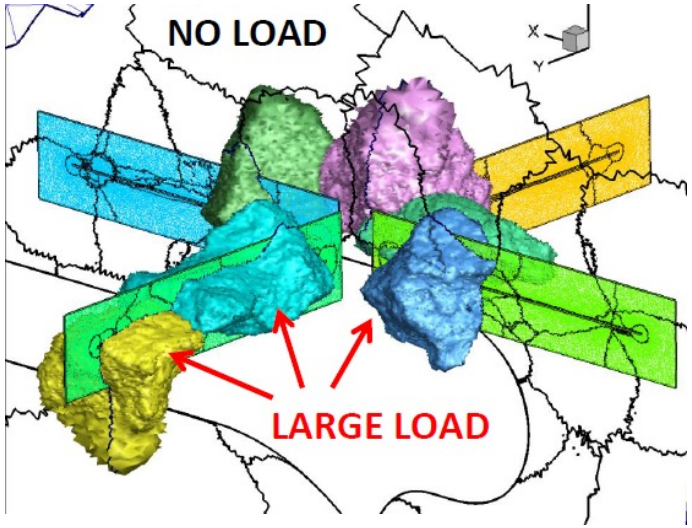
Problem: For each surface point A, find P, closest point on I_L (isosurface of distance field at L), without actually constructing the isosurface

Walk on I_L in a direction aligned with steepest descent

- Start from initial guess B
- Find closest point on surface, C
- Compute search direction as $d = (BC \times BA) \times BC$
- Take a step in direction d
- Re-compute closest point on surface: if distance up, step back and reduce step
- Repeat until convergence



Scalable Domain Connectivity



- **Traditional domain connectivity in parallel**
  - For each mesh partition, find other mesh partitions that may overlap with it (bounding box search)
  - Send points to the other mesh partition(s) and have it search for donors and report back donors

    → inevitably some mesh partitions will have more search load than others and will need active load balancing that results in diminishing returns (*Roget & Sitaraman, JCP 2014)*
- **Strand based domain connectivity**
  - Strand mesh description (surface + vectors) is available in all processors
  - Each mesh partition searches for donors for all of its points in all strand meshes, no communication is necessary
  - Improved scalability since all partitions have similar number of points by design
  - Still not perfectly scalable since search work/point will differ, but the imbalance is more moderate.