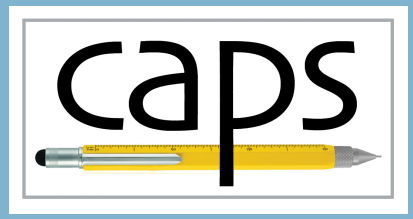
Computational Aircraft Prototype Syntheses



Training Session 6
Meshing for CFD I: AFLR

ESP v1.19

Marshall Galbraith galbramc@mit.edu
Bob Haimes haimes@mit.edu
Massachusetts Institute of Technology

John F. Dannenhoffer, III
jfdannen@syr.edu
Syracuse University

Overview

- AFLR4 surface mesh generation
  - Mesh length scaling (`capsMeshLength`)
  - Edge spacing controls
  - Proximity Detection
- AFLR3 volume mesh generation
  - Inviscid mesh generation (Parent/Child)
  - Viscous boundary layer mesh generation
- Suggested Exercises

AFLR Overview

Advancing-Front/Local-Reconnection Grid Generator

- AFLR mesh generator suite by Prof. David Marcum at Mississippi State
  - AFLR2 – 2D unstructured meshes
  - AFLR3 – 3D unstructured meshes with boundary layers
  - AFLR4 – CAD surface meshes
- AFLR2/AFLR3 developed over past decades
- AFLR4 extensive development as part of `CAPS` project
  - Actively under development

Transport Geometry EGADS Files

- Use `pyCAPS` to export geometry to EGADS files
- Explore meshing parameters without rebuilding geometry
- DANGER: Decouples geometric and analysis parameters
  - geometry.despmtr and geometry.cfgpmtr are read only

Execute: EGADS/egadsCFD.py

```
# Change to Inviscid CFD view
transport.cfgpmtr.VIEW.Concept     = 0
transport.cfgpmtr.VIEW.CFDInviscid = 1
transport.cfgpmtr.VIEW.CFDViscous  = 0

# Enable just wing
transport.cfgpmtr.COMP.Wing    = 1
transport.cfgpmtr.COMP.Fuse    = 0
transport.cfgpmtr.COMP.Htail   = 0
transport.cfgpmtr.COMP.Vtail   = 0
transport.cfgpmtr.COMP.Pod     = 0
transport.cfgpmtr.COMP.Control = 0

# Save egads file of the geometry
print("==> Generating CFDInviscid_Wing")
transport.save("CFDInviscid_Wing.egads")
```

CFDInviscid_Wing.egads
CFDInviscid_WingPod.egads
CFDInviscid_Transport.egads
CFDViscous_Wing.egads
CFDViscous_WingPod.egads
CFDViscous_Transport.egads

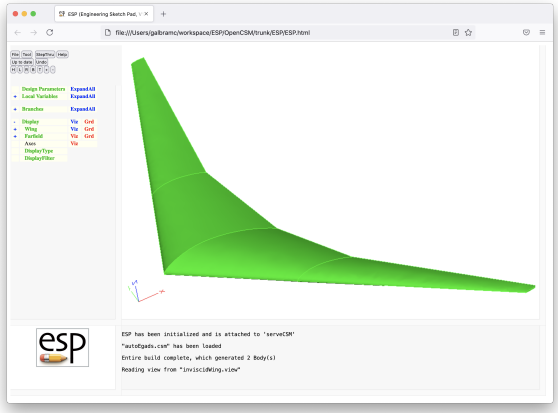
AFLR4/AFLR3 AIM Documentation

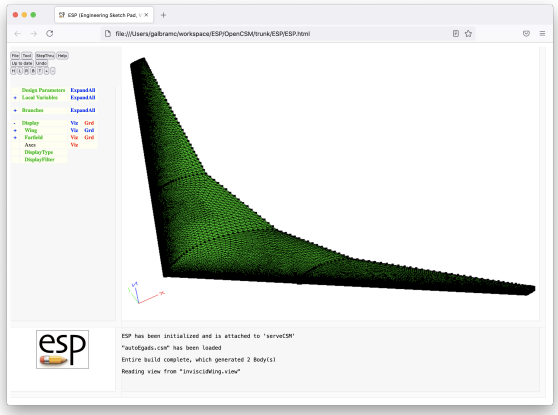
AFLR Execution and Surface Tessellations

- AFLR generates meshes in memory
- geometry.view shows surface tessellation after postAnalysis

session06/aflr4_01_InviscidWing.py

```
# Create aflr4 aim
aflr4 = myProblem.analysis.create(aim = "aflr4AIM",
                                  name = "aflr4")

# View the bodies
aflr4.geometry.view()

# Mark capsMesh == Farfield with a Farfield bcType
aflr4.input.Mesh_Sizing = {"Farfield": {"bcType":"Farfield"}}

# Run AIM pre-analysis
aflr4.preAnalysis()

# AFLR4 executes in memory with preAnalysis

# Run AIM post-analysis
aflr4.postAnalysis()

# View the surface tessellation
aflr4.geometry.view()
```





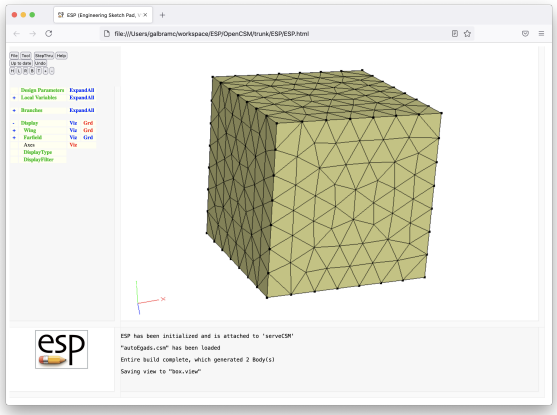
AFLR Farfield Growth Factor

- Farfield boundary set via "Mesh_Sizing", or tagged with

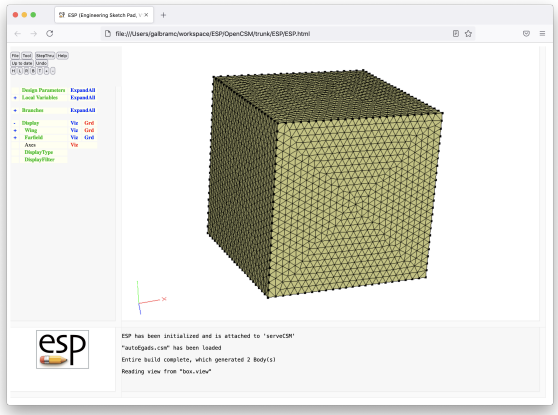
  `ATTRIBUTE AFLR_GBC $FARFIELD_UG3_GBC`

session06/aflr4_02_Farfield.py

```
# Mark capsMesh == Farfield with a Farfield bcType
aflr4.input.Mesh_Sizing = {"Farfield": {"bcType":"Farfield"}}

# Farfield growth factor
aflr4.input.ff_cdfr = 1.4
```



aflr4.input.ff_cdfr = 1.4



aflr4.input.ff_cdfr = 1.1

Mesh Length Scaling

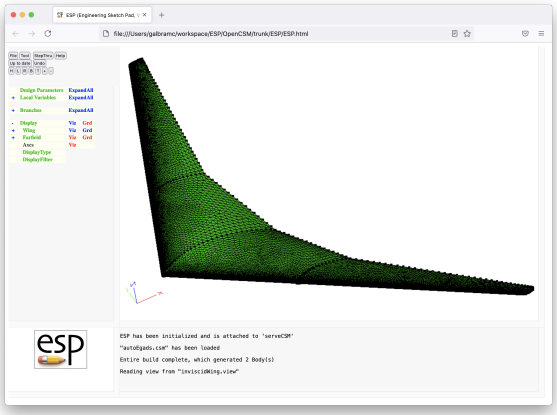
- AFLR4 reference length (ref_len) bounds element sizes
- Rule of thumb: characteristic length of geometry
  - Mean Aerodynamic Chord, diameter of fuselage, etc.
- CAPS: reference length computed from `capsMeshLength`

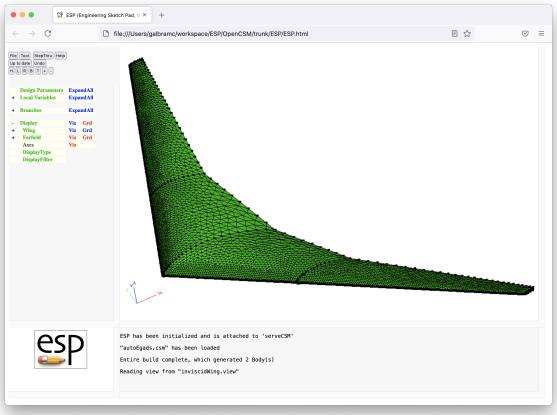
  `ATTRIBUTE capsMeshLength wing:mac`

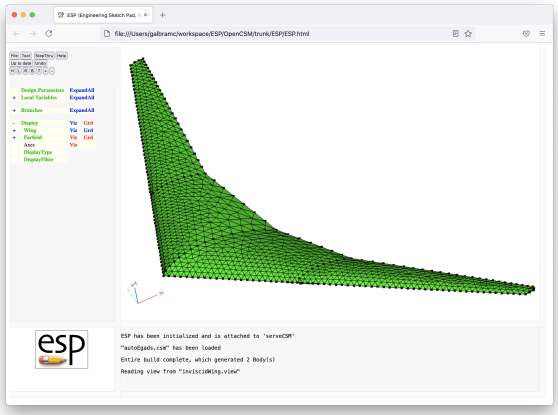
session06/aflr4_03_MeshLength.py

```
# Scaling factor to compute AFLR4 'ref_len' parameter via
# ref_len = capsMeshLength * Mesh_Length_Factor
aflr4.input.Mesh_Length_Factor = 5

# Relative scale of maximum spacing bound relative to ref_len
# max_spacing = max_scale * ref_len
aflr4.input.max_scale = 0.1

# Relative scale of minimum spacing bound relative to ref_len
# min_spacing = min_scale * ref_len
aflr4.input.min_scale = 0.01

# Absolute scale of minimum spacing bound for proximity
# abs_min_spacing = abs_min_scale * ref_len
aflr4.input.abs_min_scale = 0.001
```

Mesh Length Scaling







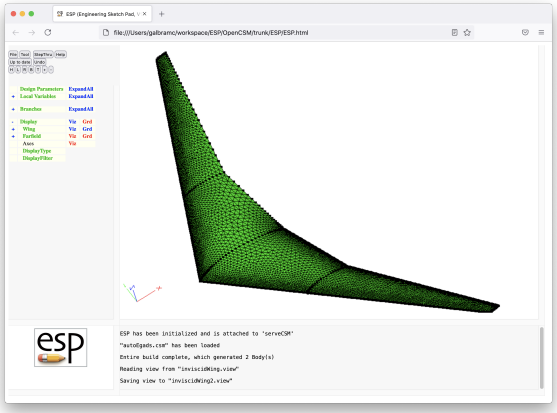
session06/aflr4_03_MeshLength.py

```
# Use mesh length factor to make a series of meshes (not a family)
for Length_Factor in [1, 3, 9]:
    aflr4.input.Mesh_Length_Factor = Length_Factor

    # Run AIM pre-/post-analysis
    aflr4.preAnalysis()
    aflr4.postAnalysis()

    # View the surface tessellation
    aflr4.geometry.view()
```

Edge Weight for Large Face-to-Face Angles

- Increase resolution of "sharp" edges (e.g. trailing edges)
- Set via "Mesh_Sizing", or attribute applied to `FACE`s

```
ATTRIBUTE AFLR4_Edge_Scale_Factor_Weight 1
```



session06/aflr4_04_EdgeWeight.py

```
# Edge mesh spacing can be scaled on surfaces based on
# discontinuity level between adjacent surfaces on both sides of the edge.
# The level of discontinuity potentially reducing the edge spacing.
# The edgeWeight scale factor weight is used as an interpolation weight
# between the unmodified spacing and the modified spacing.
aflr4.input.Mesh_Sizing = {"leftWing": {"edgeWeight":1.0},
                           "Farfield": {"bcType":"Farfield"}}
```

FACE/EDGE Local Mesh Scaling

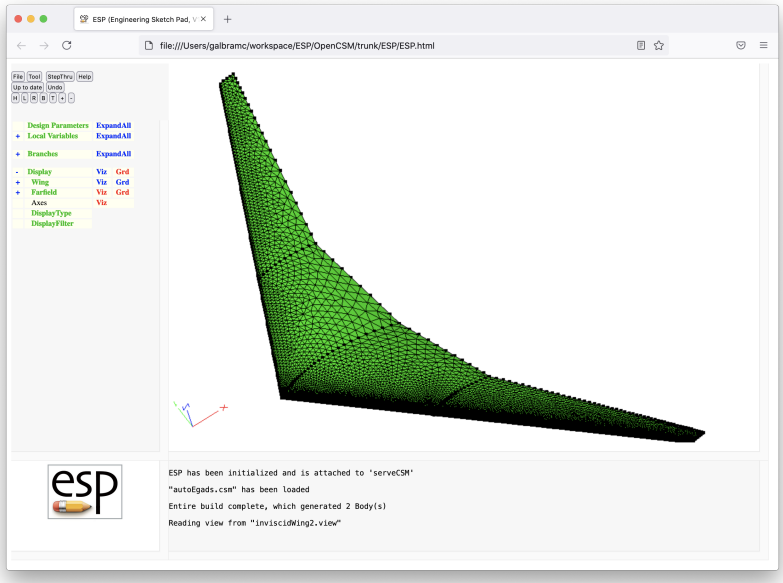
- Increase/Decrease resolution on FACE/EDGE
- Set via "Mesh_Sizing" applied ONLY to FACEs[a]

  ATTRIBUTE AFLR4_Scale_Factor 0.25

[a] AFLR4_Scale_Factor may be set on FACE/EDGE in csm script

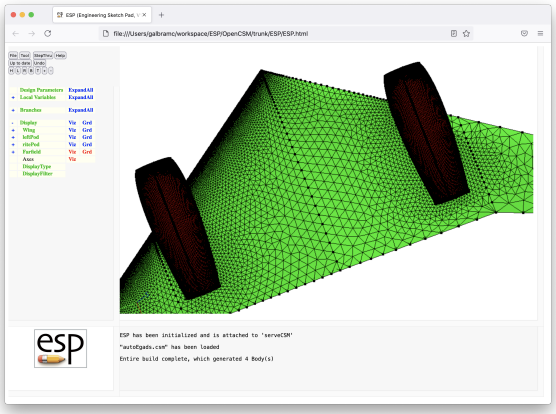
session06/aflr4_05_ScaleFactor.py

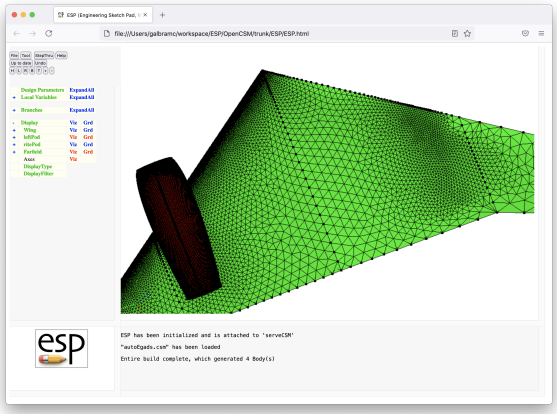
```
# Apply a local scaling factor to capsMesh="leftWing"
aflr4.input.Mesh_Sizing = {"riteWing": {"scaleFactor":2.0},
                           "leftWing": {"scaleFactor":0.5},
                           "Farfield": {"bcType":"Farfield"}}
```



Proximity Detection

- Proximity detection imprints meshes from close bodies on each other
- Wing coarsened to illustrate imprinting
- Individual bodies treated as separate components

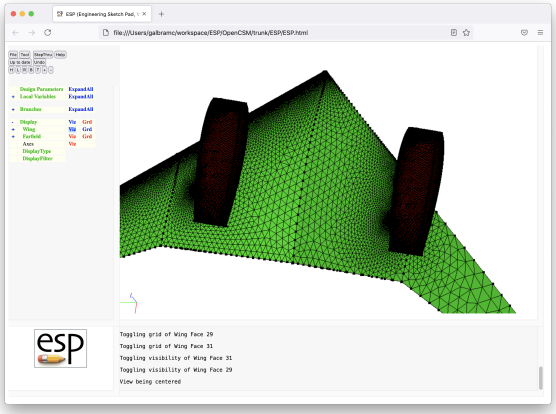


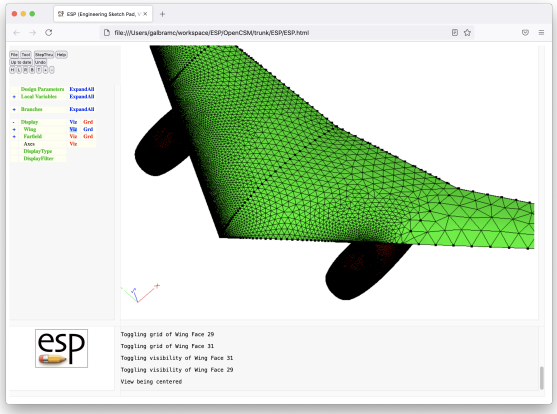


session06/aflr4_06_Proximity.py

```
# Set mesh sizing parmeters
aflr4.input.Mesh_Sizing = {"Farfield": {"bcType":"Farfield"},
                           "leftWing": {"scaleFactor":50},
                           "Pod"     : {"scaleFactor":0.5}}
```

Proximity Sensing Single Body Components





- `AFLR4_Cmp_ID FACE` attribute distinguishes components

ESP/viewCFDViscous.udc

```
SET AFLR4_Cmp_Fuse      1
SET AFLR4_Cmp_Wing      2
SET AFLR4_Cmp_Htail     3
SET AFLR4_Cmp_Vtail     4
SET AFLR4_Cmp_Pod       5
SET AFLR4_Cmp_Farfield  6
```

```
SELECT FACE
    ATTRIBUTE capsGroup    $Wing
    ATTRIBUTE AFLR4_Cmp_ID AFLR4_Cmp_Wing

UDPRIM    editAttr  filename  <<
    FACE HAS      tagComp=leftWing
    SET           capsMesh=leftWing
```

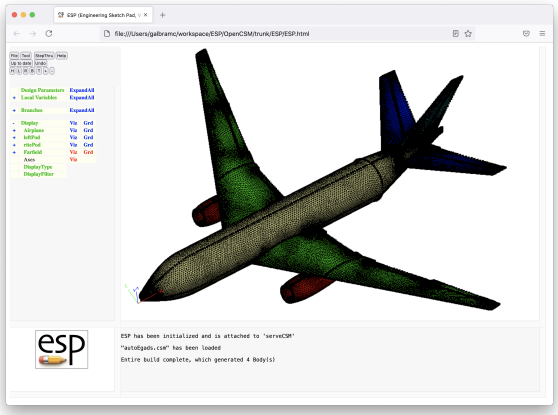
```
RESTORE   PodLeftOml
    ATTRIBUTE capsGroup    $Pod
    ATTRIBUTE capsMesh     $Pod
    ATTRIBUTE AFLR4_Cmp_ID AFLR4_Cmp_Pod
```

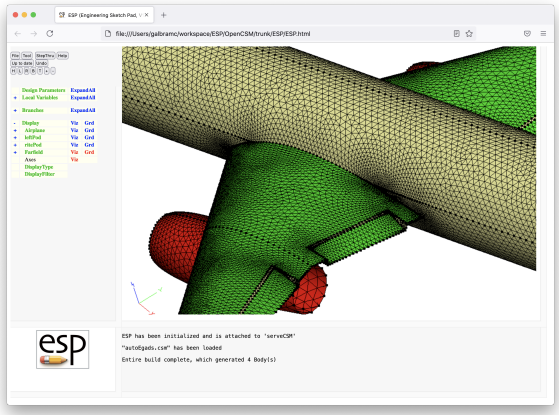
Execute: session06/aflr4_07_ProximityComponents.py

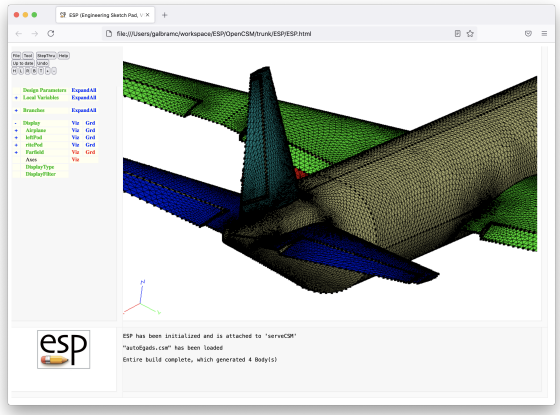
Full Inviscid Transport

- Inviscid surface mesh for the full transport configuration
  - ~10 s
  - 75k Nodes
  - 150k Triangles

Execute: session06/aflr4_08_InviscidTransport.py







Overview

- AFLR4 surface mesh generation
  - Mesh length scaling (`capsMeshLength`)
  - Edge spacing controls
  - Proximity Detection

- AFLR3 volume mesh generation
  - Inviscid mesh generation (Parent/Child)
  - Viscous boundary layer mesh generation

- Suggested Exercises

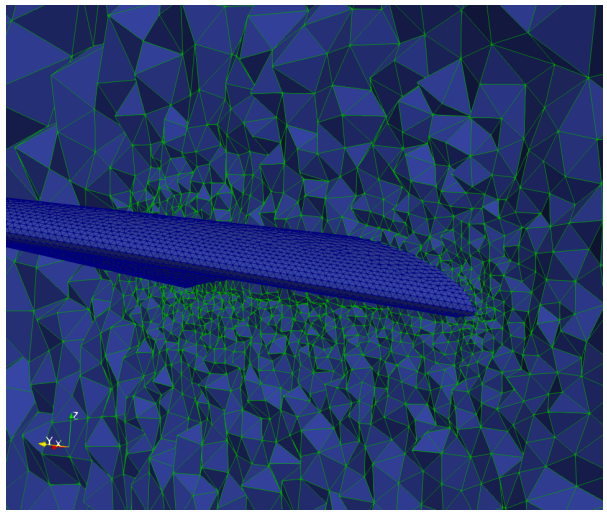
AFLR3 Volume Mesh and Parent/Child

Surface Mesh Transfer

- AFLR4 AIM output Surface_Mesh is linked to AFLR3 AIM input Surface_Mesh
- Transfers surface mesh from AFLR4 to AFLR3

session06/aflr4_aflr3_09_InviscidWing.py

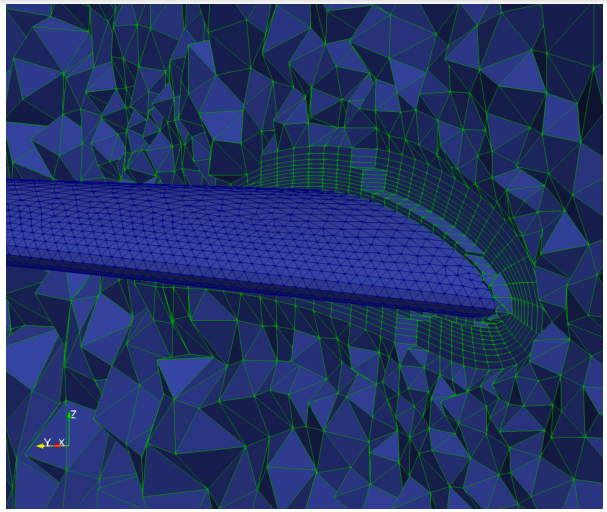
```
# Create AFLR3 AIM to generate the volume mesh
aflr3 = myProblem.analysis.create(aim  = "aflr3AIM",
                                  name = "aflr3")

# Link the aflr4 Surface_Mesh as input to aflr3
aflr3.input["Surface_Mesh"].link(aflr4.output["Surface_Mesh"])

# Dump VTK files for visualization
aflr3.input.Proj_Name   = "TransportWing"
aflr3.input.Mesh_Format = "VTK"

# Run AIM pre-/post-analysis
aflr3.preAnalysis()
aflr3.postAnalysis()
```



AFLR3 Boundary Layer Volume Mesh generation

- Boundary layer meshes are required for viscous CFD
- AFLR3 "grows" boundary layer meshes from viscous surfaces

- Global boundary layer parameter[1] (automatically ignores Farfield)
- Mesh_Gen_Input_String to add any AFLR3 command line flags



session06/aflr4_aflr3_10_ViscousWing.py

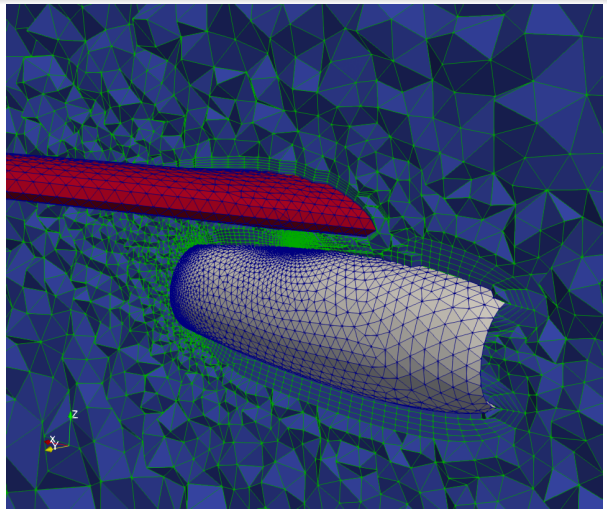
```
# Specify boundary layer maximum layers.
# Initial spacing and minimum thickness are scaled by capsMeshLength
aflr3.input.BL_Max_Layers      = 10
aflr3.input.BL_Initial_Spacing = 0.01
aflr3.input.BL_Thickness       = 0.1

# Specify prism boundary layer elements
aflr3.input.Mesh_Gen_Input_String = "-blc"
```

[1]NOTE: Unreasonably coarse boundary layers in examples

AFLR3 Boundary Layer Volume Mesh generation

- Boundary layer meshes are required for viscous CFD
- AFLR3 "grows" boundary layer meshes from viscous surfaces

- `capsGroup` wise boundary layers parameters[1]
- Properly treats colliding layers



session06/aflr4_aflr3_11_ViscousWingPod.py

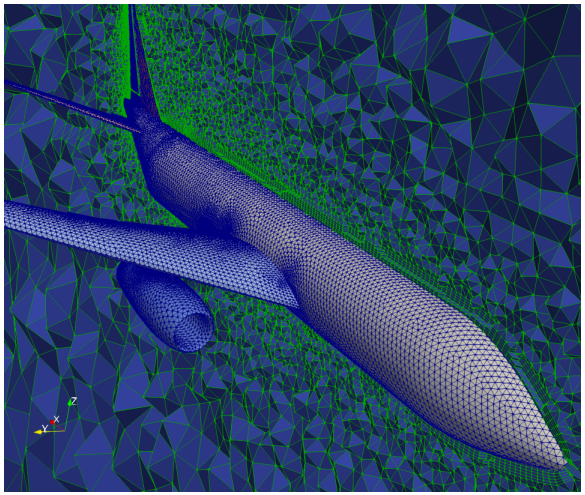
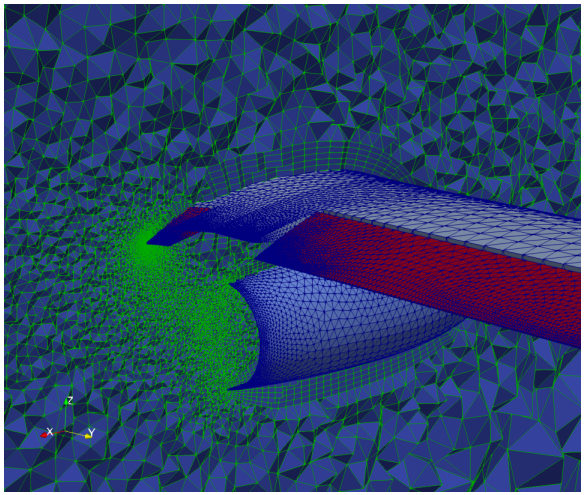
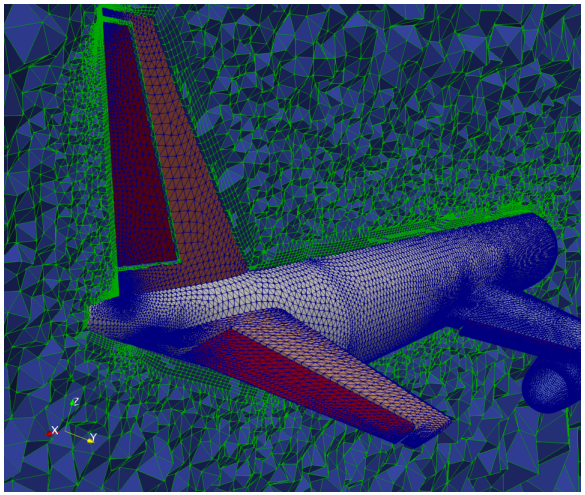
```
# Set mesh sizing parmeters
aflr3.input.Mesh_Sizing = {"leftWing" : {"boundaryLayerSpacing":0.01, "boundaryLayerThickness":0.1},
                           "riteWing" : {"boundaryLayerSpacing":0.01, "boundaryLayerThickness":0.1},
                           "Pylon"    : {"boundaryLayerSpacing":0.03, "boundaryLayerThickness":0.1},
                           "Pod"      : {"boundaryLayerSpacing":0.02, "boundaryLayerThickness":0.1}}

# Specify boundary layer maximum layers.
aflr3.input.BL_Max_Layers = 10
```

[1]NOTE: Unreasonably coarse boundary layers in examples

Full Viscous Transport

- Viscous mesh for full transport configuration[1]
  - ~2.5 min
  - 790k Nodes
  - 4.5M Elements

Execute: session06/aflr4_aflr3_12_ViscousTransport.py

[1]NOTE: Unreasonably coarse boundary layers in examples

Suggested Exercises

curv_factor

- Explore the impact of AFLR4 “curv_factor”

Inviscid Transport

- Make the surface mesh finer for the engine pods of the InviscidTransport
- Coarsen the fuselage surface mesh for the InviscidTransport without coarsening the wing/tail surfaces

Suggested Exercises Cont.

Inviscid Mesh Sequence

- For the InviscidTransport, generate surface meshes with approximate element counts of:
  - 150,000
  - 250,000
  - 300,000

- Create your own (optionally share it `galbramc@mit.edu`)