

Computational Aircraft Prototype Syntheses



Training Session 4.2 Meshing for CFD I: AFLR

Marshall Galbraith

galbramc@mit.edu

Massachusetts Institute of Technology

Bob Haines

haines@mit.edu

John F. Dannenhoffer, III

jfdannen@syr.edu

Syracuse University

- 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

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 initiated as part of CAPS project
 - Actively under development

- Use pyCAPS to export geometry to EGADS files
- Explore meshing parameters without rebuilding geometry
- DANGER: Decouples geometric and analysis parameters
 - getGeometryVal and getGeometryOutVal not available from EGADS file (available in next release)

Execute: EGADS/egadsCFD.py

```
# Change to Inviscid CFD view
transport.setGeometryVal("VIEW:Concept"      , 0)
transport.setGeometryVal("VIEW:CFDInviscid"   , 1)
transport.setGeometryVal("VIEW:CFDViscous"    , 0)
```

```
# Enable just wing
transport.setGeometryVal("COMP:Wing"          , 1)
transport.setGeometryVal("COMP:Fuse"          , 0)
transport.setGeometryVal("COMP:Htail"         , 0)
transport.setGeometryVal("COMP:Vtail"         , 0)
transport.setGeometryVal("COMP:Pod"           , 0)
transport.setGeometryVal("COMP:Control"       , 0)
```

```
# Save egads file of the geometry
transport.saveGeometry("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 generates meshes in memory
- viewGeometry shows surface tessellation after postAnalysis

session4.2/aflr4_01_InviscidWing.py

```
# Load aflr4 aim
aflr4 = myProblem.loadAIM(aim = "aflr4AIM",
                          analysisDir = "workDir_AFLR4")

# View the bodies
aflr4.viewGeometry()

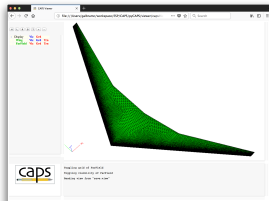
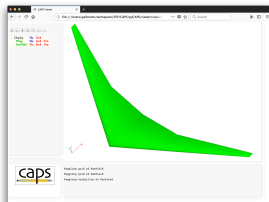
# Mark capsGroup="Farfield" with a Farfield bcType
aflr4.setAnalysisVal("Mesh_Sizing",
                    ("Farfield", {"bcType": "Farfield"}))

# Run AIM pre-analysis
aflr4.preAnalysis()

# AFLR4 executes in memory with preAnalysis

# Run AIM post-analysis
aflr4.postAnalysis()

# View the surface tessellation
aflr4.viewGeometry()
```



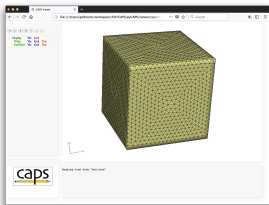
- Farfield boundary set via “Mesh_Sizing”, or tagged with attribute

ATTRIBUTE AFLR_GBC \$FARFIELD_UG3_GBC

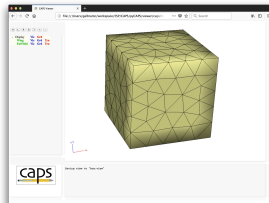
session4.2/aflr4_02_Farfield.py

```
# Mark capsGroup="Farfield" with a Farfield bcType
aflr4.setAnalysisVal("Mesh_Sizing", ("Farfield", {"bcType": "Farfield"}))
```

```
# Use 5 segments on farfield face edges
aflr4.setAnalysisVal("ff_nseg", 5)
```



Default farfield mesh



Farfield with 5 mesh segments

- AFLR4 uses a reference length to bound element sizes
- Rule of thumb: reference length characteristic length of geometry
 - Mean Aerodynamic Chord, diameter of fuselage, etc.
- CAPS: reference length computed from `capsMeshLength` attribute

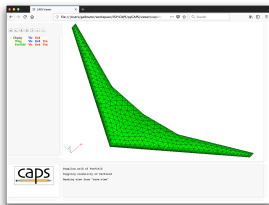
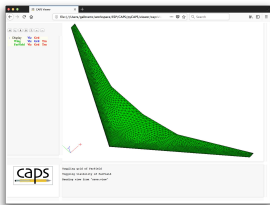
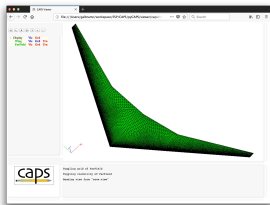
session4.2/aflr4_03_MeshLength.py

```
# Scaling factor to compute AFLR4 'ref_len' parameter via
# ref_len = capsMeshLength * Mesh_Length_Factor
aflr4.setAnalysisVal("Mesh_Length_Factor", 5)

# Relative scale of maximum spacing bound relative to ref_len
# max_spacing = max_scale * ref_len
aflr4.setAnalysisVal("max_scale", 0.1)

# Relative scale of minimum spacing bound relative to ref_len
# min_spacing = min_scale * ref_len
aflr4.setAnalysisVal("min_scale", 0.01)

# Absolute scale of minimum spacing bound for proximity
# abs_min_spacing = abs_min_scale * ref_len
aflr4.setAnalysisVal("abs_min_scale", 0.01)
```

session4.2/aflr4_03_MeshLength.py

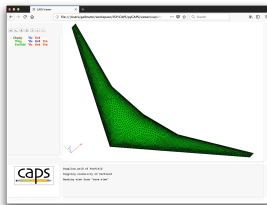
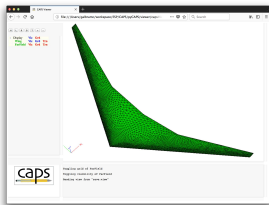
```
# Use mesh length factor to make a series of meshes (not a family)
for Mesh_Length_Factor in [1, 4, 20]:
    aflr4.setAnalysisVal("Mesh_Length_Factor", Mesh_Length_Factor)

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

    # View the surface tessellation
    aflr4.viewGeometry()
```

- Increase resolution of “sharp” edges (e.g. trailing edges)
- Set via “Mesh_Sizing”, or attribute applied to FACES

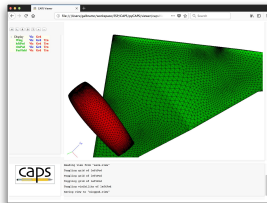
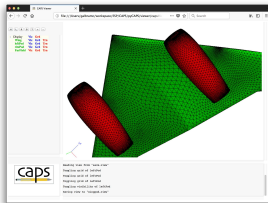
ATTRIBUTE AFLR4_Edge_Scale_Factor_Weight 1



[session4.2/aflr4_04_EdgeWeight.py](#)

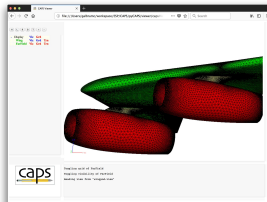
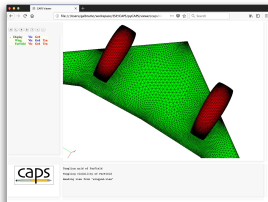
```
# Edge mesh spacing can be scaled on surfaces based on
# discontinuity level between adjacent surfaces on both sides of the edge.
# The level of discontinuity potentially reducing the edge spacing.
# The edgeWeight scale factor weight is used as an interpolation weight
# between the unmodified spacing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

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



session4.2/aflr4_05_Proximity.py

```
# Set mesh sizing parameters
aflr4.setAnalysisVal("Mesh_Sizing", [{"Farfield", {"bcType": "Farfield"}},
                                     ("Wing", {"scaleFactor": 50}),
                                     ("Pod", {"scaleFactor": 2})])
```



- 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
```

```
# put capsGroup and AFLR4_Cmp_ID on the faces
```

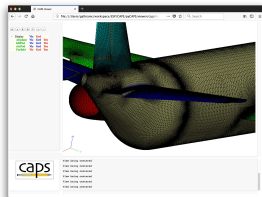
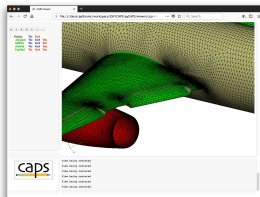
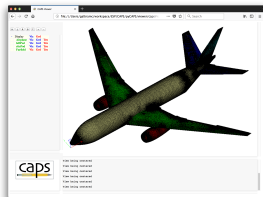
```
SELECT face
  ATTRIBUTE capsGroup    $Wing
  ATTRIBUTE AFLR4_Cmp_ID AFLR4_Cmp_Wing
```

```
RESTORE PylonLeft0ml
  ATTRIBUTE capsGroup    $Pylon
  ATTRIBUTE AFLR4_Cmp_ID AFLR4_Cmp_Pod
```

Execute: session4.2/aflr4_06_ProximityComponents.py

- Inviscid surface mesh for the full transport configuration
 - ~ 1 min
 - 110k Nodes
 - 230k Triangles

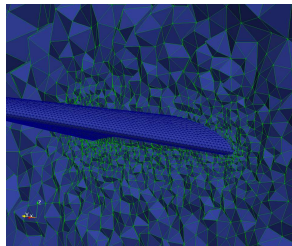
Execute: `session4.2/afr4_07_InviscidTransport.py`



- 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

Surface Mesh Transfer

- AFLR4 AIM is parent to AFLR3 AIM
- Transfers surface mesh from AFLR4 to AFLR3



session4.2/aflr4_aflr3_08_InviscidWing.py

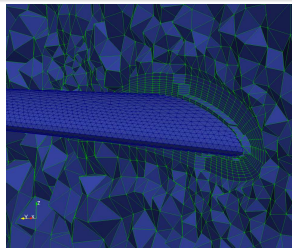
```
# Load AFLR3 AIM to generate the volume mesh
aflr3 = myProblem.loadAIM(aim           = "aflr3AIM",
                          analysisDir = "workDir_AFLR4_AFLR3_8_InviscidWing",
                          parents      = aflr4.aimName)

# Dump VTK files for visualization
aflr3.setAnalysisVal("Proj_Name", "TransportWing")
aflr3.setAnalysisVal("Mesh_Format", "VTK")

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

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

- Global boundary layer parameter¹
(automatically ignores Farfield)



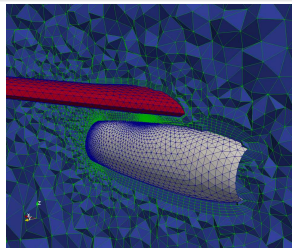
session4.2/aflr4_aflr3_09_ViscousWing.py

```
# Specify boundary layer maximum layers.  
# Initial spacing and minimum thickness are scaled by capsMeshLength  
aflr3.setAnalysisVal("BL_Max_Layers", 10)  
aflr3.setAnalysisVal("BL_Initial_Spacing", 0.01)  
aflr3.setAnalysisVal("BL_Thickness", 0.1)
```

¹NOTE: Unreasonably coarse boundary layers in examples

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

- capsGroup wise boundary layers parameters¹
- Properly treats colliding layers



session4.2/aflr4_aflr3_10_ViscousWingPod.py

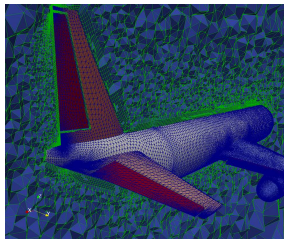
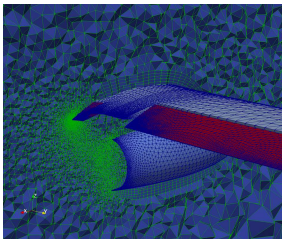
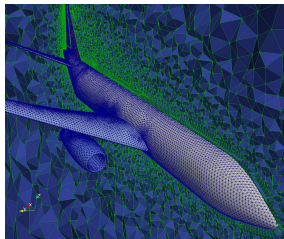
```
# Set mesh sizing parameters
aflr3.setAnalysisVal("Mesh_Sizing", [{"Wing" , {"boundaryLayerSpacing":0.01, "boundaryLayerThickness":0.1}},
                                     {"Pylon" , {"boundaryLayerSpacing":0.03, "boundaryLayerThickness":0.1}},
                                     {"Pod"   , {"boundaryLayerSpacing":0.02, "boundaryLayerThickness":0.1}}])

# Specify boundary layer maximum layers.
aflr3.setAnalysisVal("BL_Max_Layers", 5)
```

¹NOTE: Unreasonably coarse boundary layers in examples

- Viscous mesh for full transport configuration¹
 - ~ 10 min
 - 2M Nodes
 - 8M Elements

Execute: `session4.2/aflr4_aflr3_11_ViscousTransport.py`



¹NOTE: Unreasonably coarse boundary layers in examples

curv_factor

- Explore the impact of AFLR4 “curv_factor”
- Create your own