Computational Aircraft Prototype Syntheses

Training Session 6
Meshing for CFD I: AFLR

ESP v1.18

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
  - getGeometryVal and getGeometryOutVal are read only

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
print("==> Generating CFDInviscid_Wing")
transport.saveGeometry("CFDInviscid_Wing.egads")
```

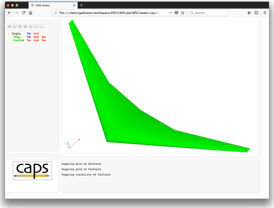
CFDInviscid_Wing.egads
CFDInviscid_WingPod.egads
CFDInviscid_Transport.egads
CFDViscous_Wing.egads
CFDViscous_WingPod.egads
CFDViscous_Transport.egads

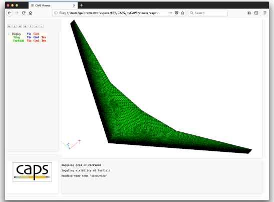
AFLR4/AFLR3 AIM Documentation

- AFLR generates meshes in memory
- viewGeometry shows surface tessellation after postAnalysis

session06/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()
```





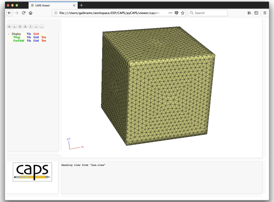
AFLR Farfield Growth Factor

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

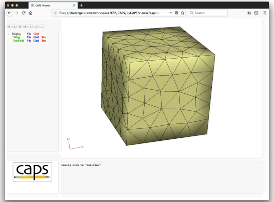
  `ATTRIBUTE AFLR_GBC $FARFIELD_UG3_GBC`

session06/aflr4_02_Farfield.py

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

# Farfield growth factor
aflr4.setAnalysisVal("ff_cdfr", 1.4)
```



Without tagging farfield FACEs



Farfield with growth factor

Mesh Length Scaling

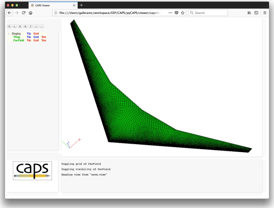
- AFLR4 reference length (ref_len) bounds element sizes
- Rule of thumb: characteristic length of geometry
  - Mean Aerodynamic Chord, diameter of fuselage, etc.
- CAPS: reference length computed from `capsMeshLength`

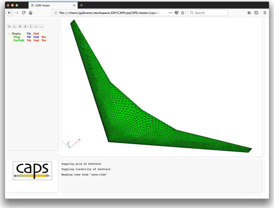
```
ATTRIBUTE capsMeshLength wing:mac
```

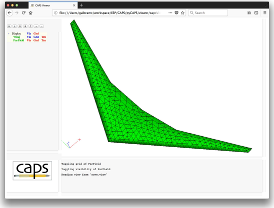
session06/aflr4_03_MeshLength.py

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

Mesh Length Scaling







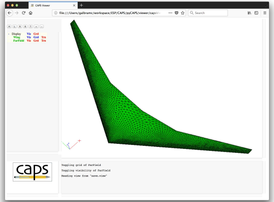
session06/aflr4_03_MeshLength.py

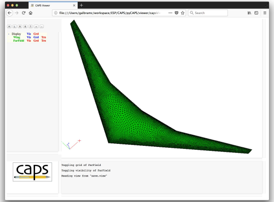
```python
# Use mesh length factor to make a series of meshes (not a family)
for Mesh_Length_Factor in [1, 3, 9]:
    aflr4.setAnalysisVal("Mesh_Length_Factor", Mesh_Length_Factor)

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

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

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
```



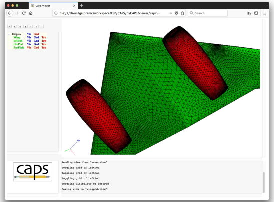


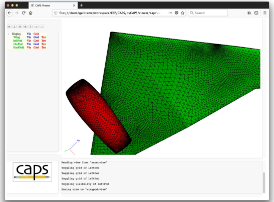
session06/aflr4_04_EdgeWeight.py

```
# 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.setAnalysisVal("Mesh_Sizing", [("Wing"    , {"edgeWeight":1.0}),
                                     ("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

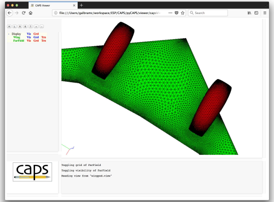


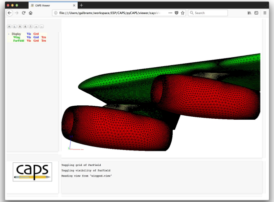


session06/aflr4_05_Proximity.py

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

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
```

```
# put capsGroup and AFR4_Cmp_ID on the faces
SELECT face
    ATTRIBUTE capsGroup    $Wing
    ATTRIBUTE AFLR4_Cmp_ID AFLR4_Cmp_Wing
```

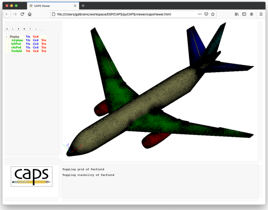
```
RESTORE   PylonLeftOml
    ATTRIBUTE capsGroup    $Pylon
    ATTRIBUTE AFLR4_Cmp_ID AFLR4_Cmp_Pod
```

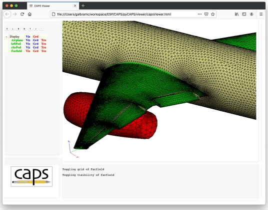
Execute: session06/aflr4_06_ProximityComponents.py

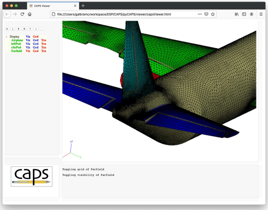
Full Inviscid Transport

- Inviscid surface mesh for the full transport configuration
  - $\sim$ 10 s
  - 95k Nodes
  - 190k Triangles

Execute: session06/aflr4_07_InviscidTransport.py







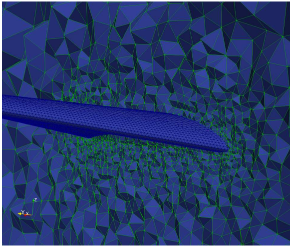
Overview

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

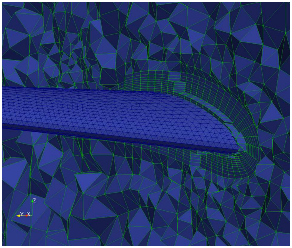


session06/aflr4_aflr3_08_InviscidWing.py

```
# 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()
```

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)

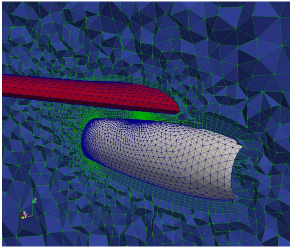
session06/aflr4_aflr3_09_ViscousWing.py

```
# 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)
```

[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_10_ViscousWingPod.py

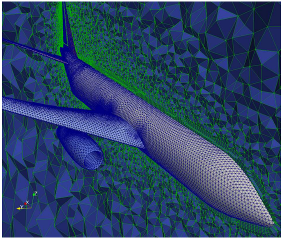
```
# Set mesh sizing parmeters
aflr3.setAnalysisVal("Mesh_Sizing",[("Wing" , {"boundaryLayerSpacing":0.01, "boundaryLayerThickness":0.1}),
                                    ("Pylon", {"boundaryLayerSpacing":0.03, "boundaryLayerThickness":0.1}),
                                    ("Pod"  , {"boundaryLayerSpacing":0.02,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

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

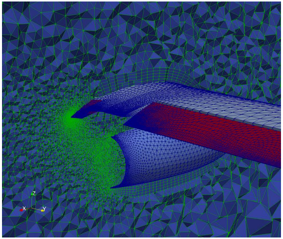
[1] NOTE: Unreasonably coarse boundary layers in examples

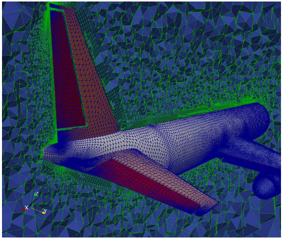
Full Viscous Transport

- Viscous mesh for full transport configuration[1]
  - ~ 2.5 min
  - 1M Nodes
  - 4.4M Elements

Execute: session06/aflr4_aflr3_11_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`)