

Computational Aircraft Prototype Syntheses



Training Session 4.3 Meshing for CFD II: Pointwise

Marshall Galbraith

galbramc@mit.edu

Massachusetts Institute of Technology

Bob Haines

haines@mit.edu

John F. Dannenhoffer, III

jfdannen@syr.edu

Syracuse University

- Pointwise and GeomToMesh.glf
- Pointwise inviscid meshing
 - Suggested parameters
 - Proximity detection
- Pointwise viscous meshing
 - Suggested parameters
 - Viscous boundary layer mesh generation
- Suggested Exercises

- Pointwise: commercial software available for Windows, Linux, and Mac.
- General purpose mesh generator for high quality structured, unstructured, and hybrid meshes
- T-Rex extrusion for generation of boundary layer resolving hybrid meshes.
- Glyph: a TCL/TK based scripting language to automate repetitive tasks.

- GeomToMesh.glf: glyph script for creating unstructured volume meshes given a clean EGADS geometry file.
- Vision: script to automatically generate a valid, high quality unstructured mesh given clean, closed geometry.
- Using geometry attributions, the scripts will attempt to create a mesh that matches the user's intent.
- Source Box for increased viscous off body resolution
- Elevate on Export generates higher-order curved meshes (not yet in CAPS)

Pointwise AIM Documentation

- 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

- Pointwise GeomToMesh.glf script available via \$CAPS_GLYPH environment variable
- Windows: also uses PW_HOME environment variable

session4.3/pointwise_01_Defaults.py

```
##### Run pointwise #####
currentDirectory = os.getcwd()    # Get current working directory
os.chdir(pointwise.analysisDir)   # Move into test directory

CAPS_GLYPH = os.environ["CAPS_GLYPH"]

if "Windows" in platform.system():
    PW_HOME = os.environ["PW_HOME"]
    os.system('"' + PW_HOME + '\\win64\\bin\\tclsh.exe ' + CAPS_GLYPH + '\\GeomToMesh.glf" caps.egads capsUserL
else:
    os.system("pointwise -b " + CAPS_GLYPH + "/GeomToMesh.glf caps.egads capsUserDefaults.glf")

os.chdir(currentDirectory)        # Move back to top directory
#####
```

- GeomToMeshDefaults.glf contains meshing parameters that control the mesh characteristics
- The default parameter settings mirror values in Pointwise interactive mode
- GeomToMesh.glf input: CAD file (egads for instance) and optional “UserDefault.glf”
- The parameters in the “UserDefaults.glf” file will override the settings in the GeomToMeshDefaults.glf

Execute: session4.3/pointwise_01_Defaults.py

- Steve Karman: “The values discussed in the following slides are, to some extent, personal preferences evolved over years of experience.”
- These parameters resolve geometry curvature and create high quality surface meshes
- The volume mesh exhibits smooth gradation of element size

session4.3/pointwise_02_InviscidWing.py

```
# Connector level
pointwise.setAnalysisVal("Connector_Turn_Angle"    , 10)
pointwise.setAnalysisVal("Connector_Source_Spacing", True)

# Domain level
pointwise.setAnalysisVal("Domain_Algorithm"        , "AdvancingFront")
pointwise.setAnalysisVal("Domain_Max_Layers"       , 15)
pointwise.setAnalysisVal("Domain_TReX_ARLimit"     , 20.0)
pointwise.setAnalysisVal("Domain_Decay"            , 0.8)

# Block level
pointwise.setAnalysisVal("Block_Boundary_Decay"    , 0.8)
pointwise.setAnalysisVal("Block_Edge_Max_Growth_Rate", 1.5)
```

- Connector attributes control the mesh operations on **EDGES**
- Geometry curve length and curvature can influence the mesh resolution and distribution
- Source Spacing enables proximity checking between connectors

session4.3/pointwise_02_InviscidWing.py

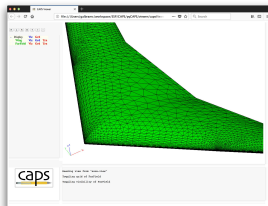
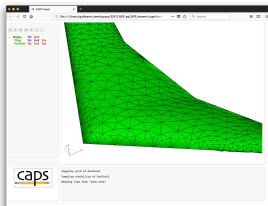
```
# Connector level
pointwise.setAnalysisVal("Connector_Turn_Angle", 10)
pointwise.setAnalysisVal("Connector_Source_Spacing", True)
```

- Domain attributes control the mesh operations on **FACES**
- Max Layers enables T-Rex surface boundary layer
 - Clustering to high curvature and sharp regions of the geometry

session4.3/pointwise_02_InviscidWing.py

```
# Domain level
```

```
pointwise.setAnalysisVal("Domain_Algorithm" , "AdvancingFront")  
pointwise.setAnalysisVal("Domain_Max_Layers" , 15)  
pointwise.setAnalysisVal("Domain_TRex_ARLimit", 20.0)  
pointwise.setAnalysisVal("Domain_Decay" , 0.8)
```



- Block attributes control the mesh operation in the volume, including the extruded viscous mesh portion

session4.3/pointwise_02_InviscidWing.py

```
# Block level
pointwise.setAnalysisVal("Block_Boundary_Decay", 0.8)
pointwise.setAnalysisVal("Block_Edge_Max_Growth_Rate", 1.5)
```

- Turning Angle resolves the surface mesh EDGES in increments to match the degree specified
- The default value for the Turning Angle Hard (70.0) will identify sharp corners in the geometry not flagged by turning angle

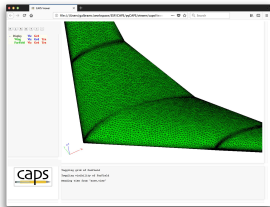
session4.3/pointwise_03_TurnAngle.py

```
# Demonstrate the impact of Connector_Turn_Angle
```

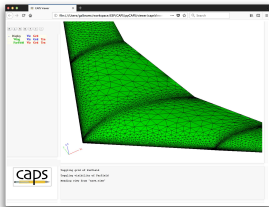
```
for conTurnAngle in [1, 10, 50]:
```

```
    # Modify the turn angle
```

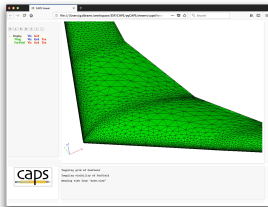
```
    pointwise.setAnalysisVal("Connector_Turn_Angle", conTurnAngle)
```



Galbraith



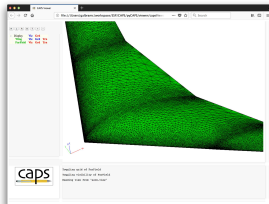
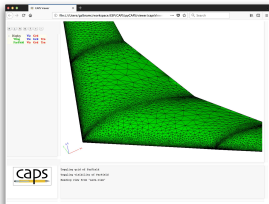
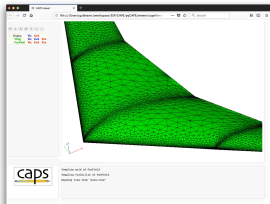
CAPS Training - Session 4.3



- Domain decay controls gradation of edge lengths away from the EDGES a surface mesh patch.
- Values near 1.0 give gradual increase in edge length.
- Values (0.5 or less) gives rapid increase in edge length.

session4.3/pointwise_04_DomainDecay.py

```
# Demonstrate the impact of Domain_Decay
for domDecay in [0.1, 0.75, 0.95]:
    # Modify the domain decay
    pointwise.setAnalysisVal("Domain_Decay", domDecay)
```

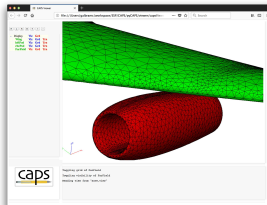
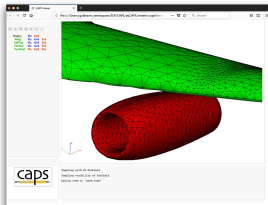


- Pod includes **EDGE** geometry close to the wing leading edge
- Source Spacing enables connectors-to-connector proximity detection
- The proximity test also uses Block level parameters

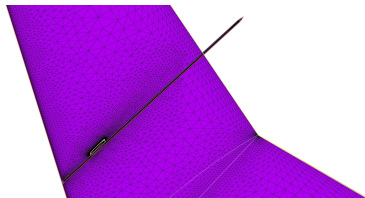
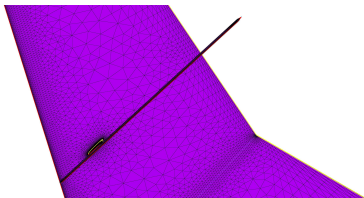
session4.3/pointwise_05_Proximity.py

```
pointwise.setAnalysisVal("Block_Boundary_Decay", 0.8)
pointwise.setAnalysisVal("Block_Edge_Max_Growth_Rate", 1.5)
```

```
# Demonstrate the impact of Connector_Source_Spacing
for conSourceSpace in [False, True]:
    # Modify the source spacing
    pointwise.setAnalysisVal("Connector_Source_Spacing", conSourceSpace)
```

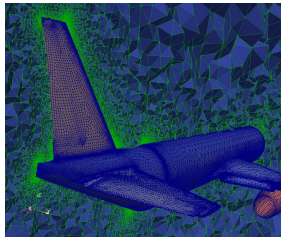
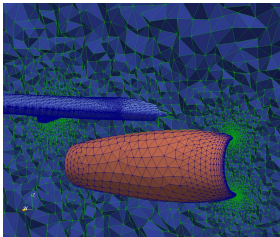
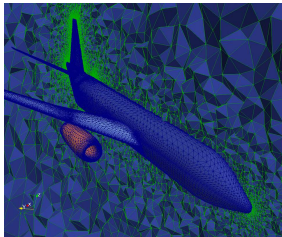


- Domain-to-domain proximity detection enabled with
Connector_Adapt_Sources
Domain_Adapt



- Inviscid surface mesh for the full transport configuration
 - ~ 5 min
 - 600k Nodes
 - 4M Elements

`session4.3/pointwise_06_InviscidTransport.py`



- Pointwise and GeomToMesh.glf
- Pointwise inviscid meshing
 - Suggested parameters
 - Proximity detection
- Pointwise viscous meshing
 - Suggested parameters
 - Viscous boundary layer mesh generation
- Suggested Exercises

- Recommended viscous values are similar to the inviscid
- Block level parameters to control boundary layer meshing

session4.3/pointwise_07_ViscousWing.py

```
pointwise.setAnalysisVal("Connector_Source_Spacing" , True)

# Domain level
pointwise.setAnalysisVal("Domain_Algorithm" , "AdvancingFront")
pointwise.setAnalysisVal("Domain_Max_Layers" , 15)
pointwise.setAnalysisVal("Domain_Growth_Rate" , 1.2)
pointwise.setAnalysisVal("Domain_TREx_ARLimit" , 20.0)
pointwise.setAnalysisVal("Domain_Decay" , 0.8)

# Block level
pointwise.setAnalysisVal("Block_Boundary_Decay" , 0.8)
pointwise.setAnalysisVal("Block_Collision_Buffer" , 1.0)
pointwise.setAnalysisVal("Block_Max_Skew_Angle" , 160.0)
pointwise.setAnalysisVal("Block_Edge_Max_Growth_Rate" , 1.5)
pointwise.setAnalysisVal("Block_Full_Layers" , 1)
pointwise.setAnalysisVal("Block_Max_Layers" , 100)

# Set wall spacing for capsGroup = Wing
viscousWall = {"boundaryLayerSpacing" : 0.001}
```

- The suggested connector attributes very similar to inviscid values.
- Proximity growth rate has been slightly reduced.

session4.3/pointwise_07_ViscousWing.py

```
pointwise.setAnalysisVal("Domain_Growth_Rate" , 1.2)
pointwise.setAnalysisVal("Domain_TReX_ARLimit", 20.0)
pointwise.setAnalysisVal("Domain_Decay"      , 0.8)

# Block level
pointwise.setAnalysisVal("Block_Boundary_Decay"      , 0.8)
```


- The suggested domain attributes also similar to inviscid values.
- The growth rate has been slightly reduced.

session4.3/pointwise_07_ViscousWing.py

```
pointwise.setAnalysisVal("Domain_Growth_Rate" , 1.2)
pointwise.setAnalysisVal("Domain_TReX_ARLimit", 20.0)
pointwise.setAnalysisVal("Domain_Decay"      , 0.8)

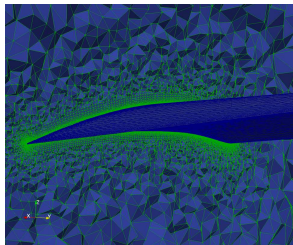
# Block level
pointwise.setAnalysisVal("Block_Boundary_Decay"      , 0.8)
```

- Increased collision buffer to allow gap between fronts
- Skew angle stops T-Rex locally when elements exceed
- Full_Layers is the desired number of full viscous layers
- Max Layer is upper bounds on the number of layers
- T-Rex automatically halts locally when elements approach isotropy

session4.3/pointwise_07_ViscousWing.py

```
# Block level
pointwise.setAnalysisVal("Block_Boundary_Decay"      , 0.8)
pointwise.setAnalysisVal("Block_Collision_Buffer"    , 1.0)
pointwise.setAnalysisVal("Block_Max_Skew_Angle"      , 160.0)
pointwise.setAnalysisVal("Block_Edge_Max_Growth_Rate", 1.5)
pointwise.setAnalysisVal("Block_Full_Layers"         , 1)
pointwise.setAnalysisVal("Block_Max_Layers"          , 100)

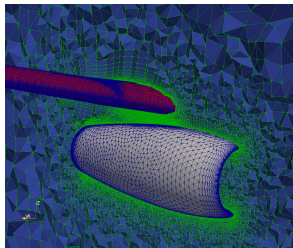
# Set wall spacing for capsGroup = Wing
viscousWall = {"boundaryLayerSpacing" : 0.001}
pointwise.setAnalysisVal("Mesh_Sizing", ("Wing", viscousWall))
```



- Viscous extrusion when "PW:WallSpacing" attribute on FACES
 - Set with Mesh_Sizing boundaryLayerSpacing using capsGroup
 - boundaryLayerSpacing scaled by capsMeshLength
- Boundary layer wall spacings can differ between capsGroup

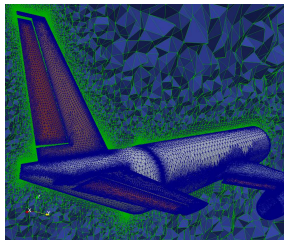
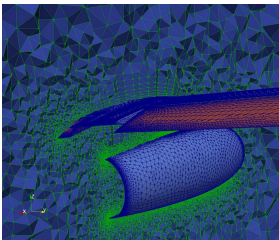
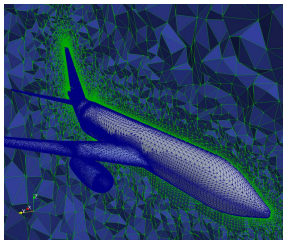
session4.3/pointwise_08_ViscousWingPod.py

```
# Set wall spacing for capsGroup = Wing and capsGroup = Pod
wingWall = {"boundaryLayerSpacing" : 0.001}
podWall  = {"boundaryLayerSpacing" : 0.003}
pointwise.setAnalysisVal("Mesh_Sizing", [{"Wing", wingWall},
                                           ("Pod", podWall)])
```



- Viscous surface mesh for the full transport configuration using suggested parameters¹
 - ~ 15 min
 - 1.8M Nodes
 - 10.2M Elements

`session4.3/pointwise_09_ViscousTransport.py`

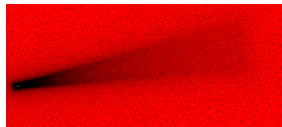
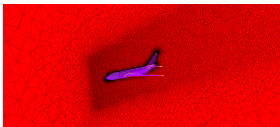


¹NOTE: Unreasonably coarse boundary layers in examples

- Box keys off of all viscous surfaces
- Creates an elongated box the direction of the vector
- Widens using the Angle parameter
- Scalar size field grows from surface size along the box

session4.3/pointwise_10_SourceBox.py

```
# General source box parameters
pointwise.setAnalysisVal("Gen_Source_Box_Length_Scale", 40.0)
pointwise.setAnalysisVal("Gen_Source_Box_Direction" , [ 0.9848077, 0, 0.1736482 ])
pointwise.setAnalysisVal("Gen_Source_Box_Angle" , 10.0)
pointwise.setAnalysisVal("Gen_Source_Growth_Factor" , 40.0)
```



Domain Proximity

- Enable domain-to-domain proximity detection for the wing with pods
 - Connector_Adapt_Sources and Domain_Adapt

Other AIM Inputs

- Explore the impact of Pointwise AIM input parameters
 - Connector, Domain, Block inputs
- Create your own