

Computational Aircraft Prototype Syntheses



Training Session 5.3

Data Transfer: Loosely-Coupled Aeroelasticity

Marshall Galbraith

galbramc@mit.edu

Massachusetts Institute of Technology

Bob Haines

haines@mit.edu

John F. Dannenhoffer, III

jfdannen@syr.edu

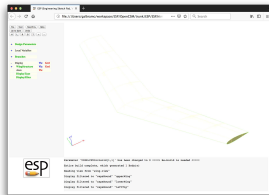
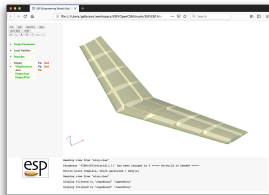
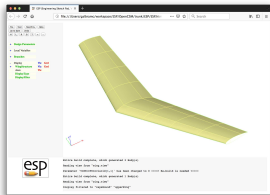
Syracuse University

- Loosely coupled analysis
- `capsBound` object and the `capsBound` attribute
 - `capsVertexSet` objects
 - `capsDataSet` objects
- Loosely coupled one-way modal aeroelastic analysis
- Loosely coupled two-way iterative aeroelastic analysis

- Couple two independent analysis tools
- Aeroelastic analysis
 - CFD to compute pressures
 - FEM to compute displacements
- Typically disparate tools
 - Mesh resolution
 - Data representation (cell vs. node center)
- CAPS data transfer reconciles differences
- Examples:
 - One-way coupling: Astros Modal \rightarrow Fun3D
 - Two-way coupling: Astros Static \leftrightarrow SU2

- capsBound is a logical grouping of BRep Objects
 - Represent the same entity, e.g. “outer surface of the wing”
- Bound is used by CAPS framework to facilitate data transfer
 - Defined by the capsBound attribute
- Same capsBound attribute applied to “coincident” bodies defines connection

ESP/wing3.csm Structures capsBound



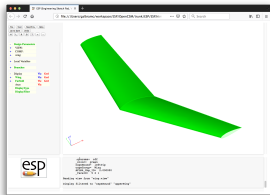
ATTRIBUTE capsBound \$SupperWing

ATTRIBUTE capsBound \$lowerWing

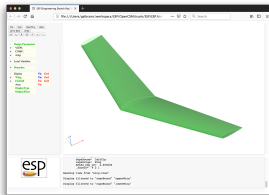
ATTRIBUTE capsBound \$leftTip

- capsBound is a logical grouping of BRep Objects
 - Represent the same entity, e.g. “outer surface of the wing”
- Bound is used by CAPS framework to facilitate data transfer
 - Defined by the capsBound attribute
- Same capsBound attribute applied to “coincident” bodies defines connection

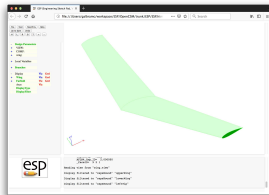
ESP/wing3.csm CFD Inviscid capsBound



ATTRIBUTE capsBound \$upperWing



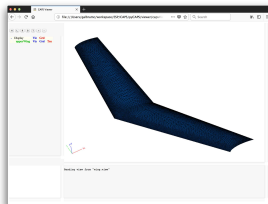
ATTRIBUTE capsBound \$lowerWing



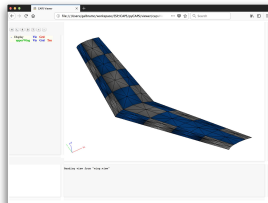
ATTRIBUTE capsBound \$leftTip

- capsBound capsVertexSet

capsVertexSet



Aerodynamic



Structures

- capsVertexSet is a discrete capsBound
- If capsBound Faces have same underlying surface, then the native UVs are used to match the points between capsVertexSets
- If not the triangulations are reparameterized with a single UV representation
- If barycentric coordinates are found for each vertex in a capsVertexSet to the other(s). This allows for straight interpolation using the solver's spatial discretization scheme (as provided in the AIM)

capsDataSet

- Discrete data associated with a capsVertexSet
 - Pressure
 - Temperature
 - Displacements
- Flexible data structure
 - Node centered data
 - Cell centered data
 - Higher-order nodal basis functions

- Data transferred between different `capsVertexSet`
 - Pressure from aero to structures `capsVertexSet`
 - Displacements from structures to aero `capsVertexSet`
- Transfer via interpolation:
 - Interpolation, does not insure integrated values match between `capsVertexSets` – important for a convergent inner loop
- Conservative transfer:
 - Conservative data transfers ensure integrated quantities match by slightly adjusting (weighting) the interpolation.

- Loosely coupled analysis
- capsBound object and the capsBound attribute
 - capsVertexSet objects
 - capsDataSet objects
- Loosely coupled one-way modal aeroelastic analysis
- Loosely coupled two-way iterative aeroelastic analysis

- Compute EigenVectors with Astros
- Transfer EigenVectors to Fun3D for aeroelastic calculation

session5.3/aeroelastic_Modal_Fun3D_Astros.py

- Load aflr4AIM
- Load aflr3AIM
- Load fun3dAIM
- Load egadsTessAIM
- Load astrosAIM
- Create capsBound data transfers
- Generate meshes
- Fill capsVertexSet
- Execute ASTROS
- Transfer EigenVectors from ASTROS to Fun3D
- Execute Fun3D

- variableName: Variable names
- aimSrc: AIM names to transfer data from (source)
- aimDest: AIM names receiving data (destination)
- transferMethod: Interpolate or Conserve
- capsBound: Name of the capsBound attribute on the bodies
- Builds dictionary myProblem.dataBound[bound]

session5.3/aeroelastic_Modal_Fun3D_Astros.py

```
# Create an array of EigenVector names
numEigenVector = 3
eigenVector = []
for i in range(numEigenVector):
    eigenVector.append("EigenVector_" + str(i+1))

# Create the capsBounds for data transfer
transfers = ["upperWing", "lowerWing", "leftTip", "riteTip"]
for bound in transfers:
    myProblem.createDataTransfer(variableName = eigenVector,
                                aimSrc       = [astros.aimName]*numEigenVector,
                                aimDest      = [fun3d.aimName]*numEigenVector,
                                transferMethod = ["Conserve"]*numEigenVector,
                                capsBound    = bound)
```

- Generate meshes with pre/post Analysis
- Populate vertex sets

session5.3/aeroelastic_Modal_Fun3D_Astros.py

```
# Run AIM pre/post-analysis to generate the meshes
for aim in [aflr4.aimName, aflr3.aimName, tess.aimName]:
    myProblem.analysis[aim].preAnalysis()
    myProblem.analysis[aim].postAnalysis()

#-----#

# Populate vertex sets in the bounds after the mesh generation is completed
for bound in transfers:
    myProblem.dataBound[bound].fillVertexSets()
```

- Execute Astros
- Transfer each EigenVector for each capsBound

session5.3/aeroelastic_Modal_Fun3D_Astros.py

```
#Execute the dataTransfer
print ("\nExecuting dataTransfer .....")
for bound in transfers:
    for eigenName in eigenVector:
        myProblem.dataBound[bound].executeTransfer(eigenName)
```

- Execute Fun3D

- Loosely coupled analysis
- capsBound object and the capsBound attribute
 - capsVertexSet objects
 - capsDataSet objects
- Loosely coupled one-way modal aeroelastic analysis
- Loosely coupled two-way iterative aeroelastic analysis

- Compute pressures with SU2
- Compute displacements with ASTROS
- Displace CFD mesh, and compute pressures with SU2

session5.3/aeroelastic_Iterative_SU2_Astros.py

- Load aflr4AIM
- Load aflr3AIM
- Load su2AIM
- Load egadsTessAIM
- Load astrosAIM
- Create capsBound data transfers
- Generate meshes
- Fill capsVertexSet
- Iterate
 - Transfer displacements from ASTROS to SU2
 - Execute SU2
 - Transfer pressure from SU2 to ASTROS
 - Execute ASTROS

- Interleave AIM names in aimSrc and aimDest
- Initial value applied to Displacement to start iterations

session5.3/aeroelastic_Iterative_SU2_Astros.py

```
# Create the data transfer connections
transfers = ["upperWing", "lowerWing", "leftTop", "riteTip"]
for bound in transfers:
    myProblem.createDataTransfer(variableName = ["Pressure", "Displacement"],
                                aimSrc       = [su2.aimName, astros.aimName],
                                aimDest      = [astros.aimName, su2.aimName],
                                transferMethod = ["Conserve", "Interpolate"],
                                initValueDest = [None, (0,0,0)],
                                capsBound    = bound )
```

- Generating mesh and capsVertexSet

- Start iterations

session5.3/aeroelastic_Iterative_SU2_Astros.py

```
# Aeroelastic iteration loop
for iter in range(numTransferIteration):

    #Execute the dataTransfer of displacements to su2
    #initValueDest is used on the first iteration
    print ("\n\nExecuting dataTransfer \"Displacement\".....")
    for bound in transfers:
        myProblem.dataBound[bound].executeTransfer("Displacement")
```

- Execute SU2

```
#Execute the dataTransfer of Pressure to astros
print ("\n\nExecuting dataTransfer \"Pressure\".....")
for bound in transfers:
    myProblem.dataBound[bound].executeTransfer("Pressure")
```

- Execute ASTROS