

Computational Aircraft Prototype Syntheses



Training Session 10

Structures Analysis: ASTROS/NASTRAN

ESP v1.18

Marshall Galbraith

galbramc@mit.edu

Massachusetts Institute of Technology

Bob Haines

haines@mit.edu

John F. Dannenhoffer, III

jfdannen@syr.edu

Syracuse University

- Modal analysis using ASTROS
 - Cantilever
 - Support node
- Static analysis using ASTROS
 - Cantilever
 - Orthotropic materials and coordinate systems
- Flutter analysis with NASTRAN

Automated Structural Optimization System

- ZONA Technology, Inc.
 - Originally developed by Northrop Corporation under contract with AF Wright Aeronautical Laboratories
- Structural Modal and Static Analysis
- Aerodynamic Loads (Vortex Lattice Method)
- Aeroelastic Stability and Trim
- Control System Interaction
- Structural Sizing Optimization
- Sensitivity Analysis

micro-ASTROS (mASTROS)

- Limited mesh sizes
- No aerodynamic analysis

NASA STRucture ANalysis

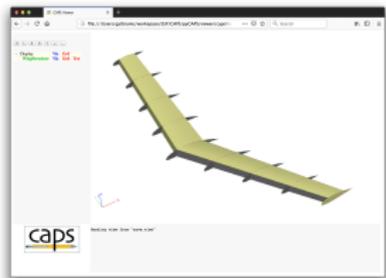
- MSC Software Corporation (MSC Nastran)
 - Originally developed for NASA in late 1960s by MSC
 - AutoDesk NEi Software (NEi Nastran)
 - Siemens PLM Software (NX Nastran)
 - Open source (<https://github.com/nasa/NASTRAN-95>)
- Structural Modal and Static Analysis
- Aerodynamic Loads (Vortex Lattice Method)
- Aeroelastic Stability and Trim
- Structural Assembly Modeling
- Automated Structural Optimization

ASTROS AIM Documentation

- Box structure with spars and ribs
- Cantilever root constraint
- Support node at root

ESP/wing3.csm

```
CFGPMTR VIEW:ClampedStructure 0
CFGPMTR VIEW:SupportStructure 0
CFGPMTR VIEW:BoxStructure 0
```



- Use egadsTessAIM for quad surface tessellation
- Coarsest possible grid for expedience
- Consistent for all examples in this session

session10/astros_1_ModalClamped.py

```
# Load EGADS tess aim
tess = myProblem.loadAIM(aim = "egadsTessAIM",
                        analysisDir = "workDir_Tess")

# No Tess vertexes on edges (minimal mesh)
tess.setAnalysisVal("Edge_Point_Max", 2)

# Use regularized quads
tess.setAnalysisVal("Mesh_Elements", "Quad")

# Run AIM pre/post-analysis to generate the surface mesh
print ("\n=> Generating Mesh...")
tess.preAnalysis()
tess.postAnalysis()
```

- Cantilever constraint on wing root rib
FACE/EDGE/NODE using capsConstraint

ESP/viewStructure.udc

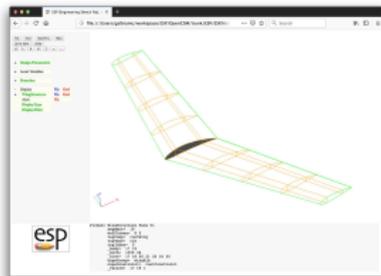
```
# Constrained applied to FACE/EDGE/NODE of root rib
```

```
UDPRIM   editAttr filename <<
FACE HAS   tagComp=rootWing
SET       capsConstraint=rootConstraint
```

```
EDGE ADJ2FACE tagComp=rootWing
SET   capsConstraint=rootConstraint
```

```
NODE ADJ2FACE tagComp=rootWing
SET   capsConstraint=rootConstraint
```

>>



session10/astros_1_ModalClamped.py

```
wing.setGeometryVal("VIEW:Concept"          , 0)
wing.setGeometryVal("VIEW:ClampedStructure", 1)
wing.setGeometryVal("VIEW:BoxStructure"     , 1)
```

```
# Set constraints
constraint = {"groupName" : ["rootConstraint"],
             "dofConstraint" : 123456}
astros.setAnalysisVal("Constraint", ("rootConstraint", constraint))
```

- Specify the type of analysis
- Build material database

session10/astros_1_ModalClamped.py

```
# Set analysis type
eigen = { "analysisType"      : "Modal",
          "extractionMethod"  : "SINV",
          "frequencyRange"    : [0, 10],
          "numEstEigenvalue"   : 1,
          "numDesiredEigenvalue" : 10,
          "eigenNormaliztion"  : "MASS"}
astros.setAnalysisVal("Analysis", ("EigenAnalysis", eigen))

# Set materials
unobtainium = {"youngModulus" : 2.2E6 ,
               "poissonRatio" : .5,
               "density"       : 7850}
madeupium   = {"materialType"  : "isotropic",
               "youngModulus"  : 1.2E5 ,
               "poissonRatio"  : .5,
               "density"       : 7850}
astros.setAnalysisVal("Material", [("Unobtainium", unobtainium),
                                   ("Madeupium", madeupium)])
```

- Define shell properties that will be associated with capsGroup
- Properties connected to materials via their name

session10/astros_1_ModalClamped.py

```
# Set properties
skinShell = {"propertyType" : "Shell",
             "material"      : "unobtainium",
             "bendingInertiaRatio" : 1.0,
             "shearMembraneRatio" : 0, # Turn of shear - no materialShear
             "membraneThickness" : 0.05}
ribShell = {"propertyType" : "Shell",
            "material"      : "unobtainium",
            "bendingInertiaRatio" : 1.0,
            "shearMembraneRatio" : 0, # Turn of shear - no materialShear
            "membraneThickness" : 0.1}
sparShell = {"propertyType" : "Shell",
             "material"      : "madeupium",
             "bendingInertiaRatio" : 1.0,
             "shearMembraneRatio" : 0, # Turn of shear - no materialShear
             "membraneThickness" : 0.2}
```

- Associate shell properties with capsGroup

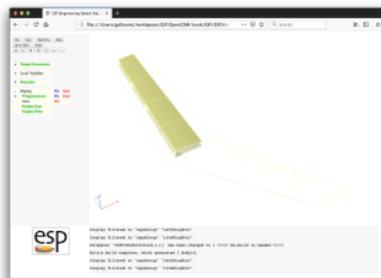
ESP/viewStructure.ude

```
# Identify the capsGroups for FEA properties
UDPRIM   editAttr  filename  <<
FACE HAS      tagComp=leftWing
ANDNOT ADJ2EDGE tagType=leadingEdge
ANDNOT ADJ2EDGE tagType=trailingEdge
ANDNOT ADJ2FACE tagType=trailingEdge
SET         capsGroup=leftWingSkin

FACE HAS      tagComp=riteWing
ANDNOT ADJ2EDGE tagType=leadingEdge
ANDNOT ADJ2EDGE tagType=trailingEdge
ANDNOT ADJ2FACE tagType=trailingEdge
SET         capsGroup=riteWingSkin
```

session10/astros_1_ModalClamped.py

```
astros.setAnalysisVal("Property", [
    ("leftWingSkin", skinShell),
    ("riteWingSkin", skinShell),
    ("wingRib"      , ribShell),
    ("wingSpar1"   , sparShell),
    ("wingSpar2"   , sparShell)])
```



- Associate shell properties with capsGroup

ESP/viewStructure.udc

```
FACE HAS      tagType=rib
SET           capsGroup=wingRib
```

```
FACE HAS      tagType=tip
SET           capsGroup=wingRib
```

session10/astros_1_ModalClamped.py

```
astros.setAnalysisVal("Property", [{"leftWingSkin", skinShell},
                                   ("riteWingSkin", skinShell),
                                   ("wingRib", ribShell),
                                   ("wingSpar1", sparShell),
                                   ("wingSpar2", sparShell)])
```



- Associate shell properties with capsGroup

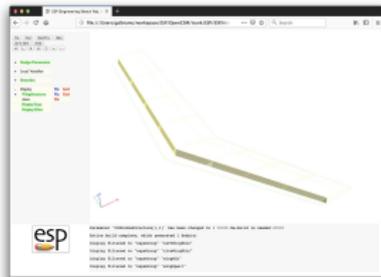
ESP/viewStructure.udc

```
FACE HAS      tagType=spar tagIndex=1
SET           capsGroup=wingSpar1
```

```
FACE HAS      tagType=spar tagIndex=2
SET           capsGroup=wingSpar2
```

session10/astros_1_ModalClamped.py

```
astros.setAnalysisVal("Property", [("leftWingSkin", skinShell),
                                    ("riteWingSkin", skinShell),
                                    ("wingRib"      , ribShell),
                                    ("wingSpar1"    , sparShell),
                                    ("wingSpar2"    , sparShell)])
```



- ASTROS requires “ASTRO.D01” and “ASTRO.IDX” in run directory

session10/astros_1_ModalClamped.py

```
astrosInstallDir = os.environ['ESP_ROOT'] + os.sep + "bin" + os.sep

# Copy files needed to run astros
files = ["ASTRO.D01", "ASTRO.IDX"]
for file in files:
    try:
        shutil.copy2(astrosInstallDir + file, file)
    except:
        print ('Unable to copy "' + file + '"')
        raise

# Run micro-ASTROS via system call
os.system("mastros.exe < " + Proj_Name + ".dat > " + Proj_Name + ".out")

# Remove temporary files
for file in files:
    if os.path.isfile(file):
        os.remove(file)
```

- Print the Eigen frequencies

session10/astros_1_ModalClamped.py

```
# Get list of Eigen-frequencies
freqs = astros.getAnalysisOutVal("EigenFrequency")

print ("\n--> Eigen-frequencies:")
for i in range(len(freqs)):
    print ("    " + repr(i+1).ljust(2) + ": " + str(freqs[i]))
```

- Support connection on wing root rib FACE/EDGE/NODE using capsConnectLink and capsConnect

ESP/viewStructure.ude

Connections to support on FACE/EDGE/NODE of root rib

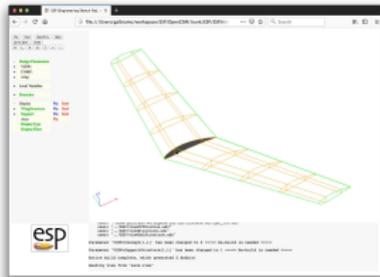
```
UDPRIM   editAttr filename <<
FACE HAS   tagComp=rootWing
SET       capsConnectLink=ribRoot
```

```
EDGE ADJ2FACE tagComp=rootWing
SET   capsConnectLink=ribRoot
```

```
NODE ADJ2FACE tagComp=rootWing
SET   capsConnectLink=ribRoot
```

Point Connecting to ribRoot to apply boundary conditions

```
POINT wing:xroot+wing:chordr/4 wing:yroot wing:zroot
ATTRIBUTE capsConnect      $ribRoot
ATTRIBUTE capsGroup        $ribSupport
ATTRIBUTE capsConstraint    $ribRootPoint
```



session10/astros_2_ModalSupport.py

```
wing.setGeometryVal("VIEW:Concept"      , 0)
wing.setGeometryVal("VIEW:SupportStructure", 1)
wing.setGeometryVal("VIEW:BoxStructure"  , 1)
```

- Define connection and support types
- Define constraint on the support node

ESP/viewStructure.udc

```
# Point Connecting to ribRoot to apply boundary conditions
POINT wing:xroot+wing:chordr/4 wing:yroot wing:zroot
ATTRIBUTE capsConnect      $ribRoot
ATTRIBUTE capsGroup        $ribSupport
ATTRIBUTE capsConstraint    $ribRootPoint
```

session10/astros_2_ModalSupport.py

```
# Defined Connections
connection = {"dofDependent" : 123456,
             "connectionType" : "RigidBody"}
astros.setAnalysisVal("Connect", ("ribRoot", connection))

# Set supports
support = {"groupName" : ["ribRootPoint"],
          "dofSupport": 3}
astros.setAnalysisVal("Support", ("ribSupport", support))

# Set constraints
constraint = {"groupName" : ["ribRootPoint"],
             "dofConstraint" : 12456}
astros.setAnalysisVal("Constraint", ("ribConstraint", constraint))
```

- Define concentrated mass with moments of inertia on support node

ESP/viewStructure.udc

```
# Point Connecting to ribRoot to apply boundary conditions
POINT wing:xroot+wing:chordr/4 wing:yroot wing:zroot
ATTRIBUTE capsConnect      $ribRoot
ATTRIBUTE capsGroup        $ribSupport
ATTRIBUTE capsConstraint    $ribRootPoint
```

session10/astros_2_ModalSupport.py

```
mass = {"propertyType" : "ConcentratedMass",
        "mass"         : 1.0E5,
        #I11 I12 I22   I31 I32 I33
        "massInertia" : [0.0, 0.0, 1.0E5, 0.0, 0.0, 0.0]}

astros.setAnalysisVal("Property", [(("leftWingSkin", skinShell),
                                     ("riteWingSkin", skinShell),
                                     ("wingRib"      , ribShell),
                                     ("wingSpar1"    , sparShell),
                                     ("wingSpar2"    , sparShell),
                                     ("ribSupport"   , mass      )])])
```

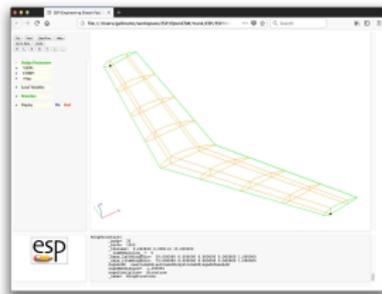
- Modal analysis using ASTROS
 - Cantilever
 - Support node
- Static analysis using ASTROS
 - Cantilever
 - Orthotropic materials and coordinate systems
- Flutter analysis with NASTRAN

- Static loads can be applied to entities marked with capsLoad
- capsLoad on NODE → point load
- capsLoad on EDGE → multi-point load (future linear load)
- capsLoad on FACE → pressure load

ESP/viewStructure.udc

```
# Define the point(s) at which point loads can be applied
UDPRIM   editAttr filename <<
NODE ADJ2FACE tagComp=leftWing tagType=tip
AND ADJ2FACE tagType=lower
AND ADJ2FACE tagType=spar tagIndex=1
SET      capsLoad=leftPointLoad

NODE ADJ2FACE tagComp=riteWing tagType=tip
AND ADJ2FACE tagType=lower
AND ADJ2FACE tagType=spar tagIndex=1
SET      capsLoad=ritePointLoad
```



session10/astros_3_StaticClamped.py

```
wing.setGeometryVal("VIEW:Concept"          , 0)
wing.setGeometryVal("VIEW:ClampedStructure", 1)
wing.setGeometryVal("VIEW:BoxStructure"     , 1)
```

- Define loads to apply, and set analysis to static

ESP/viewStructure.udc

```
# Define the point(s) at which point loads can be applied
UDPRIM   editAttr filename <<
         NODE ADJ2FACE tagComp=leftWing tagType=tip
         AND  ADJ2FACE tagType=lower
         AND  ADJ2FACE tagType=spar tagIndex=1
         SET   capsLoad=leftPointLoad

         NODE ADJ2FACE tagComp=riteWing tagType=tip
         AND  ADJ2FACE tagType=lower
         AND  ADJ2FACE tagType=spar tagIndex=1
         SET   capsLoad=ritePointLoad
```

session10/astros_3_StaticClamped.py

```
# Define loads
leftLoad = {"loadType"      : "GridForce",
            "forceScaleFactor" : 1.e6,
            "directionVector" : [0.0, 0.0, 1.0]}
riteLoad = {"loadType"      : "GridForce",
            "forceScaleFactor" : 2.e6,
            "directionVector" : [0.0, 0.0, 1.0]}
astros.setAnalysisVal("Load", [{"leftPointLoad", leftLoad },
                               {"ritePointLoad", riteLoad }])

# Set analysis type
astros.setAnalysisVal("Analysis_Type", "Static")
```

- Print the displacements

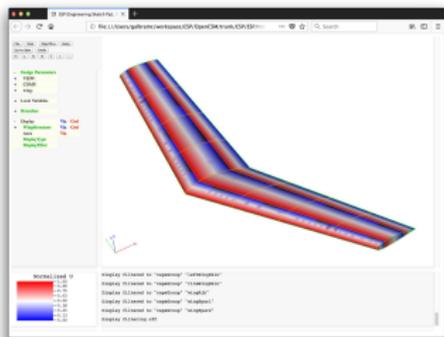
session10/astros_3_StaticClamped.py

```
print ("\n--> Maximum displacements:")
print ("--> Tmax" , astros.getAnalysisOutVal("Tmax" ))
print ("--> T1max", astros.getAnalysisOutVal("T1max"))
print ("--> T2max", astros.getAnalysisOutVal("T2max"))
print ("--> T3max", astros.getAnalysisOutVal("T3max"))
```

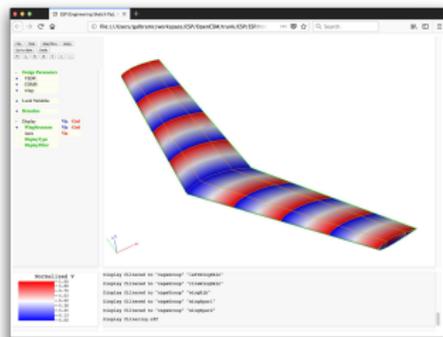
- Coordinate system name matches capsGroup

ESP/viewStructure.udc

```
# Apply the Csystems
# Name matches capsGroup that uses the Csystem
RESTORE WingStruct      #iface;ubar0;vbar0;du2;dv2
  CSYSTEM leftWingSkin  leftWingSkinCsys;0;0;0;1
  CSYSTEM riteWingSkin  riteWingSkinCsys;0;0;0;1
```



Normalized u-coordinates

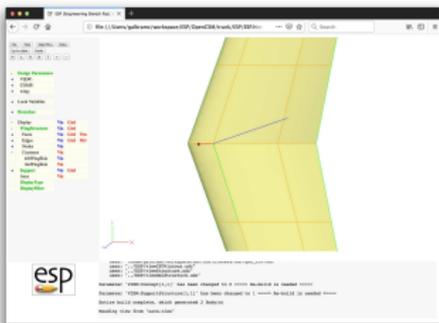


Normalized v-coordinates

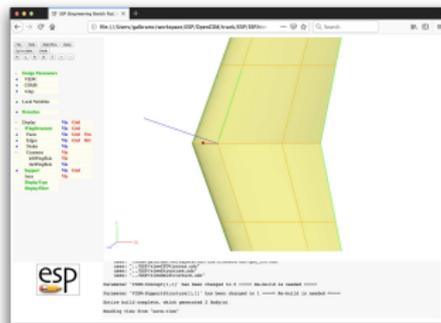
- Coordinate system name matches capsGroup

ESP/viewStructure.udc

```
# Apply the Csystems
# Name matches capsGroup that uses the Csystem
RESTORE WingStruct #iface;ubar0;vbar0;du2;dv2
CSYSTEM leftWingSkin leftWingSkinCsys;0;0;0;1
CSYSTEM riteWingSkin riteWingSkinCsys;0;0;0;1
```



leftWingSkin coordinate system



riteWingSkin coordinate system

- Define isotropic and orthotropic materials

session10/astros_4_Composite.py

```
# Set materials
Aluminum = {"youngModulus" : 10.5E6 ,
            "poissonRatio" : 0.3,
            "density"       : 0.1/386,
            "shearModulus"  : 4.04E6}

Graphite_epoxy = {"materialType"      : "Orthotropic",
                  "youngModulus"      : 20.8E6 ,
                  "youngModulusLateral" : 1.54E6,
                  "poissonRatio"      : 0.327,
                  "shearModulus"      : 0.80E6,
                  "density"            : 0.059/386,
                  "tensionAllow"      : 11.2e-3,
                  "tensionAllowLateral" : 4.7e-3,
                  "compressAllow"     : 11.2e-3,
                  "compressAllowLateral" : 4.7e-3,
                  "shearAllow"        : 19.0e-3,
                  "allowType"         : 1}

astros.setAnalysisVal("Material", [{"Aluminum", Aluminum},
                                   ("Graphite_epoxy", Graphite_epoxy)])
```

● CompositeOrientation angles relative to CSYSTEM

session10/astros_4_Composite.py

```
# Set properties
skinShell = {"propertyType"      : "Composite",
             "shearBondAllowable" : 1.0e6,
             "bendingInertiaRatio" : 1.0,
             "shearMembraneRatio" : 0, # Turn off shear - no materialShear
             "compositeMaterial"  : ["Graphite_epoxy"]*8,
             "compositeThickness" : [0.00525]*8,
             "compositeOrientation" : [0, 0, 0, 0, -45, 45, -45, 45],
             "symmetricLaminate"  : True,
             "compositeFailureTheory" : "STRAIN" }

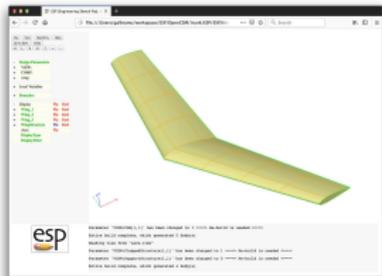
ribShell = {"propertyType"      : "Shell",
            "material"          : "Aluminum",
            "bendingInertiaRatio" : 1.0,
            "shearMembraneRatio" : 0, # Turn of shear - no materialShear
            "membraneThickness"  : 0.125 }

sparShell = {"propertyType"      : "Shell",
             "material"          : "Aluminum",
             "bendingInertiaRatio" : 1.0,
             "shearMembraneRatio" : 0, # Turn of shear - no materialShear
             "membraneThickness"  : 0.125 }

astros.setAnalysisVal("Property", [("leftWingSkin", skinShell),
                                   ("riteWingSkin", skinShell),
                                   ("wingRib", ribShell),
                                   ("wingSpar1", sparShell),
                                   ("wingSpar2", sparShell)])
```

- Modal analysis using ASTROS
 - Cantilever
 - Support node
- Static analysis using ASTROS
 - Cantilever
 - Orthotropic materials and coordinate systems
- Flutter analysis with NASTRAN

- Cantilever aeroelastic flutter analysis

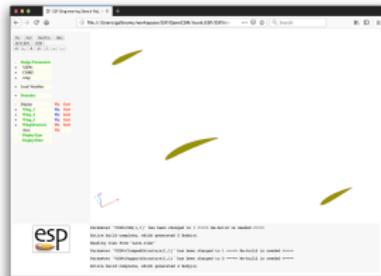


Structural body with full skin

```
ATTRIBUTE capsAIM $nastranAIM
ATTRIBUTE capsDiscipline $Structure
```

session10/nastran_5_Flutter.py

```
# Change to Structures and VLM
wing.setGeometryVal("VIEW:Concept"      , 0)
wing.setGeometryVal("VIEW:VLM"          , 1)
wing.setGeometryVal("VIEW:ClampedStructure", 1)
```



VLM Aero bodies

```
ATTRIBUTE capsAIM $nastranAIM
ATTRIBUTE capsDiscipline $Aerodynamic
```

- Aeroelastic analysis coupled via capsBound

ESP/viewStructure.udc

```
# Set up for possible data transfer to other analyses
UDPRIM   editAttr filename <<
FACE HAS   tagType=upper
SET       capsBound=upperWing

FACE HAS   tagType=lower
SET       capsBound=lowerWing

FACE HAS   tagType=tip tagIndex=1
SET       capsBound=leftTip
```

session10/nastran_5_Flutter.py

```
# Set analysis type
nastran.setAnalysisVal("Analysis_Type", "AeroelasticFlutter")

# Aero with capsGroup for airfoil sections
wingVLM = {"groupName"      : "Wing",
           "numChord"       : 4,
           "numSpanPerSection" : 6}

# Note the surface name corresponds to the capsBound found in the *.csm file. This links
# the spline for the aerodynamic surface to the structural model
nastran.setAnalysisVal("VLM_Surface", ("upperWing", wingVLM))
```

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