

# Computational Aircraft Prototype Syntheses



## Training Session 2

### CAPS Analysis

ESP v1.26

**Marshall Galbraith**

[galbramc@mit.edu](mailto:galbramc@mit.edu)

Massachusetts Institute of Technology

**Bob Haimes**

[haimes@mit.edu](mailto:haimes@mit.edu)

**John F. Dannenhoffer, III**

[jfdannen@syr.edu](mailto:jfdannen@syr.edu)

Syracuse University

- AIM creation and accessing/modifying analysis values
  - analysis.input
- Analysis execution and outputs
  - pre/postAnalysis
  - analysis.output
- Directing bodies to AIMS
  - Attributes `capsAIM` and `capsIntent`
- `capsGroup` attribute
  - Connecting geometry with analysis properties
- Suggested Exercises

- AIM creation and accessing/modifying analysis values
  - analysis.input
- Analysis execution and outputs
  - pre/postAnalysis
  - analysis.output
- Directing bodies to AIMs
  - Attributes capsAIM and capsIntent
- capsGroup attribute
  - Connecting geometry with analysis properties
- Suggested Exercises

## XFOIL

- Airfoil performance analysis
- Integral Boundary Layer Theory + Potential Panel Method
  - Subsonic flow (Prandtl–Glauert)
- $e^N$  Transition Model



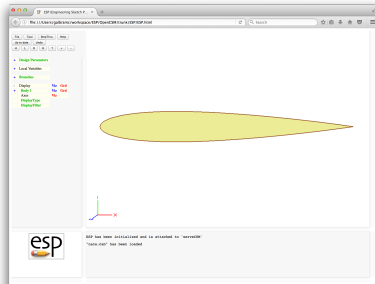
session02/naca.csm

---

```
# NACA design paramters
DESPMTR  thick      0.12      #frac of local chord
DESPMTR  camber     0.00      #frac of local chord

# Construct the airfoil
UDPRIM   naca      Thickness thick  Camber  camber
ATTRIBUTE cpaAIM $xfoilAIM;mseAIM;tsfoilAIM
```

---



- Analysis AIM created with Problem.analysis.create
  - Problem.analysis is a Sequence of Analysis Objects
  - create method adds Analysis instance to Problem.analysis Sequence
- “aim” is the string AIM type
- “name” is the unique key Problem.analysis Sequence

## session02/01\_xfoil\_input.py

```
# Load geometry [.csm] file
filename = "naca.csm"
print ("\n==> Loading geometry from file \" "+filename+"\"...")
capsProblem = pyCAPS.Problem(problemName = "design_naca",
                             capsFile = filename,
                             outLevel = 1)

# Create xfoil aim
print ("\n==> Creating xfoilAIM")
xfoil = capsProblem.analysis.create(aim = "xfoilAIM",
                                   name = "xfoil")

# Analysis added to capsProblem.analysis Sequence
assert(xfoil == capsProblem.analysis["xfoil"])
```

- Analysis inputs are set/accessed with Analysis.input Sequence of Value Objects

session02/01\_xfoil\_input.py

```
# Set Mach number
xfoil.input.Mach = 0.5

# Print the modified mach number
mach = xfoil.input.Mach
print("\n==> Modified Mach =", mach)
```

- Analysis values can be tuples/lists and toggled

## session02/01\_xfoil\_input.py

```
# Print the default value of None
print("\n==> Default Alpha =", xfoil.input["Alpha"].value)

# Set Alpha number
xfoil.input.Alpha = 2.5
print("\n==> Modified Alpha =", xfoil.input["Alpha"].value)

# Set list of Alpha
xfoil.input.Alpha = [0.0, 3.0, 5.0, 7.0, 8.0]
print("\n==> Modified Alpha =", xfoil.input["Alpha"].value)

# Unset Alpha back to None
xfoil.input.Alpha = None
print("\n==> Unset Alpha =", xfoil.input["Alpha"].value)
print()
```

- Available analysis input values in xfoil AIM documentation

## xfoil AIM Documentation

- AIM creation and accessing/modifying analysis values
  - analysis.input
- Analysis execution and outputs
  - pre/postAnalysis
  - analysis.output
- Directing bodies to AIMS
  - Attributes capsAIM and capsIntent
- capsGroup attribute
  - Connecting geometry with analysis properties
- Suggested Exercises

- Create AIM and set analysis values
  - autoExec used to toggle automatic execution when available

## session02/02\_xfoil\_Analysis.py

---

```
# Create xfoil aim
print ("\n==> Creating xfoilAIM")
xfoil = capsProblem.analysis.create(aim = "xfoilAIM",
                                   name = "xfoil",
                                   autoExec = False)

print ("\n==> Setting analysis values")
# Set Mach and Reynolds number
xfoil.input.Mach = 0.5
xfoil.input.Re   = 1.0e6

# Set list of Alpha
xfoil.input.Alpha = [0.0, 3.0, 5.0, 7.0, 8.0]
```

---

- Run preAnalysis to generate xfoil input files

session02/design\_naca/Scratch/xfoil/caps.xfoil

session02/design\_naca/Scratch/xfoil/xfoilInput.txt

- “Scratch” is the default Phase
- “xfoil” in the path is the AIM name

session02/02\_xfoil\_Analysis.py

```
print ("\n=> Creating xfoilAIM")
xfoil = capsProblem.analysis.create(aim = "xfoilAIM",
                                    name = "xfoil",
                                    autoExec = False)
```

```
# Run AIM pre-analysis
print ("\n=> Running preAnalysis")
xfoil.preAnalysis()
```

- Execute xfoil in session02/design\_naca/Scratch/xfoil
- Use Analysis.system method for system call!
  - Executes in xfoil.analysisDir
  - Required for “Journaling/Continuation”

## session02/02\_xfoil\_Analysis.py

---

```
# Run xfoil via system call
print ("\n\n==> Running xFoil.....")
xfoil.system("xfoil < xfoilInput.txt > Info.out");
```

---

- Run postAnalysis to indicate completion and parse output files

session02/02\_xfoil\_Analysis.py

```
# Run AIM post-analysis
print ("\n==> Running postAnalysis")
xfoil.postAnalysis()
```

- Get outputs with analysis.output Value Object Sequence

```
# Retrieve Alpha, Cl and Cd
print ("\n==> Retrieve analysis results")
Alpha = xfoil.output.Alpha
Cl     = xfoil.output.Cl
Cd     = xfoil.output.Cd

print()
print("--> Alpha =", Alpha)
print("--> Cl    =", Cl)
print("--> Cd    =", Cd)
print()
```

- Create AIM with autoExec = True (default)
- Explicitly execute with runAnalysis (optional)

### session02/03\_xfoil\_AutoExec.py

```
# Create xfoil aim
print ("\n=> Creating xfoilAIM")
xfoil = capsProblem.analysis.create(aim = "xfoilAIM",
                                    name = "xfoil",
                                    autoExec = True)
```

```
# Run analysis (optional)
xfoil.runAnalysis()
```

- Auto execution available for many AIMs
- Not implemented for analysis with MPI
- Auto execution used throughout rest of the training

- Setup analysis values

session02/04\_xfoil\_Camber.py

---

```
print ("\n==> Setting analysis values")
# Set Mach and Reynolds number
xfoil.input.Mach = 0.5
xfoil.input.Re   = 1.0e6

# Set list of Alpha
xfoil.input.Alpha = [0.0, 1.0, 3.0]
```

---

- Execute sequence of cambers

## session02/04\_xfoil\_Camber.py

---

```
# List of cambers to analyze
Camber = [0.00, 0.01, 0.04, 0.07]

Alpha = []; Cl = []; Cd = []
for camber in Cambers:
    # Modify the camber
    naca.despmtr.camber = camber

    # Append Alpha, Cl and Cd
    print ("\n==> Retrieve analysis results. camber = ", naca.despmtr.camber)
    Alpha.append(xfoil.output.Alpha)
    Cl.append(xfoil.output.CL)
    Cd.append(xfoil.output.CD)

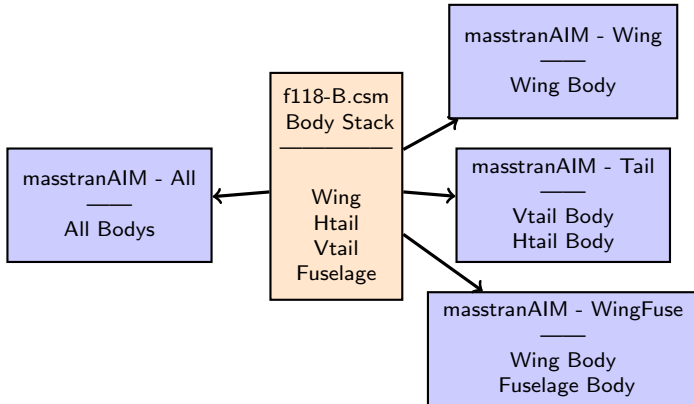
print()
print("--> Cambers =", Cambers)
print("--> Alpha  =", Alpha)
print("--> Cl     =", Cl)
print("--> Cd     =", Cd)
print()
```

---

- AIM creation and accessing/modifying analysis values
  - analysis.input
- Analysis execution and outputs
  - pre/postAnalysis
  - analysis.output
- Directing bodies to AIMS
  - Attributes `capsAIM` and `capsIntent`
- capsGroup attribute
  - Connecting geometry with analysis properties
- Suggested Exercises

# Directing Bodies to AIMs

- **masstranAIM:** Computes mass properties via a shell mesh
  - Compute mass properties for collections of bodies
- Single Stack with all Bodies
- Direct bodies in Stack to different AIM instances



- capsAIM BODY attribute
  - String semicolon separated AIM names
  - AIMs suitable to use the body
- capsIntent BODY attribute
  - Optional string used to direct bodies to AIM instance
  - String semicolon separated names
  - Multiple bodies may have the same capsIntent

session02/f118-B.csm

```
SET      htail:span      sqrt(htail:aspect*htail:area)
SET      htail:chord     htail:area/htail:span

BOX      htail:xroot    -htail:span/2  htail:zroot  htail:chord  htail:span  htail:chord*htail:thick
SELECT  body
  ATTRIBUTE capsAIM      $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent   $htail;tail
  ATTRIBUTE _name        $Htail
```

- Create AIM with analysis.create
  - aim: The type of the AIM to create
  - capsIntent: List of strings to match with `capsIntent` attr
  - name: A unique name (key in analysis Sequence)
- Empty `capsIntent` loads all bodies with matching `capsAIM`

### session02/05\_masstran\_AIM.py

---

```
# capsAIM == $masstranAIM
masstranAll = capsProblem.analysis.create(aim = "masstranAIM", capsIntent="",
                                          name = "All")

# The AIM instance is also available in the Problem.analysis Sequence
assert(masstranAll == capsProblem.analysis["All"])

# Show the geometry used by the AIM
print("==> Geometry used by masstranAll instance with no capsIntent")
masstranAll.geometry.view()
```

---

- Creating masstranAIM with bodies

capsAIM == **\$masstranAIM** and capsIntent == **\$wing**

session02/f118-B.csm

```
SET      wing:span      sqrt(wing:aspect*wing:area)
SET      wing:chord      wing:area/wing:span

BOX      wing:xroot      -wing:span/2      wing:zroot      wing:chord      wing:span      wing:chord*wing:thick
SELECT    body
  ATTRIBUTE capsAIM      $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent    $wing
  ATTRIBUTE _name         $Wing
```

session02/05\_masstran\_AIM.py

```
# capsAIM == $masstranAIM and capsIntent == $wing
capsProblem.analysis.create(aim = "masstranAIM", capsIntent="wing",
                           name = "Wing")

# Show the geometry used by the AIM
print("==> Geometry used by Wing instance with capsIntent='wing'")
capsProblem.analysis["Wing"].geometry.view()
```

- Creating masstranAIM with bodies

capsAIM == **\$masstranAIM** and capsIntent == **\$tail**

session02/f118-B.csm

```
BOX htail:xroot -htail:span/2 htail:zroot htail:chord htail:span htail:chord*htail:thick
SELECT body
  ATTRIBUTE capsAIM $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent $htail;tail
  ATTRIBUTE _name $Htail
```

```
BOX vtail[i,4] vtail[i,5]-vtail:thick/2 vtail[i,6] vtail:chord vtail:thick vtail:span
SELECT body
  ATTRIBUTE capsAIM $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent $vtail;tail
  ATTRIBUTE _name $Vtail+i
```

session02/05\_masstran\_AIM.py

```
# capsAIM == $masstranAIM and capsIntent == $tail
masstranTail = capsProblem.analysis.create(aim = "masstranAIM", capsIntent="tail",
                                           name = "Tail")
```

```
# Show the geometry used by the AIM
print("==> Geometry used by Tail instance with capsIntent='tail'")
masstranTail.geometry.view()
#-----#
```

- Creating masstranAIM with bodies  
`capsAIM == $masstranAIM` and  
`(capsIntent == $wing or capsIntent == $fuse)`

session02/f118-B.csm

```
BOX wing:xroot -wing:span/2 wing:zroot wing:chord wing:span wing:chord*wing:thick
SELECT body
  ATTRIBUTE capsAIM $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent $wing
  ATTRIBUTE _name $Wing
```

```
BOX 0 -fuse:width/2 -fuse:height/2 fuse:length fuse:width fuse:height
SELECT body
  ATTRIBUTE capsAIM $masstranAIM;astrosAIM
  ATTRIBUTE capsIntent $fuse
  ATTRIBUTE _name $Fuselage
```

session02/05\_masstran\_AIM.py

```
# capsAIM == $masstranAIM and (capsIntent == $wing or capsIntent == $fuse)
capsProblem.analysis.create(aim = "masstranAIM", capsIntent=["wing","fuse"],
                           name = "WingFuse")
```

- AIM creation and accessing/modifying analysis values
  - analysis.input
- Analysis execution and outputs
  - pre/postAnalysis
  - analysis.output
- Directing bodies to AIMS
  - Attributes capsAIM and capsIntent
- **capsGroup** attribute
  - Connecting geometry with analysis properties
- Suggested Exercises

## capsGroup attribute

- Tags groups of BODY/FACE/EDGE/NODE
  - Entities with same capsGroup value are in the group
- Specific use of capsGroup is AIM dependent

### session02/06\_masstran\_f118\_Wing.py

```
filename = "f118-C.csm"
print ("\n==> Loading geometry from file \""+filename+"\"...")
capsProblem = pyCAPS.Problem(problemName = "design_f118",
                             capsFile = filename,
                             outLevel = 0)

# Create a masstran aim with the wing
masstran = capsProblem.analysis.create(aim = "masstranAIM",
                                       name = "masstran",
                                       capsIntent="wing")
```

- Wing FACES tagged with `$wing:faces`
- masstranAIM material and properties for `"wing:faces"`

session02/f118-C.csm

```
BOX wing:xroot -wing:span/2 wing:zroot wing:chord wing:span wing:chord*wing:thick
SELECT face
ATTRIBUTE capsGroup $wing:faces
```

session02/06\_masstran\_f118\_Wing.py

```
# Define material properties
unobtainium = {"density" : 7850}

# Set the material
masstran.input.Material = {"Unobtainium": unobtainium}

# Define shell property
shell = {"propertyType"      : "Shell",
         "material"          : "Unobtainium",
         "membraneThickness" : 0.2}

# Associate the shell property with capsGroups defined on the geometry
masstran.input.Property = {"wing:faces": shell}
```

- capsGroups on all FACES

session02/f118-C.csm

```
BOX wing:xroot -wing:span/2 wing:zroot wing:chord wing:span wing:chord*wing:thick
SELECT face
    ATTRIBUTE capsGroup $wing:faces
```

```
BOX htail:xroot -htail:span/2 htail:zroot htail:chord htail:span htail:chord*htail:thick
SELECT face
    ATTRIBUTE capsGroup $htail:faces
```

```
SELECT face
    ATTRIBUTE capsGroup $vtail:faces
SELECT body
```

```
SELECT face 1
    ATTRIBUTE capsGroup $fuse:nose
SELECT face 2
    ATTRIBUTE capsGroup $fuse:tail
SELECT face 3
    ATTRIBUTE capsGroup $fuse:side
SELECT face 4
    ATTRIBUTE capsGroup $fuse:side
SELECT face 5
    ATTRIBUTE capsGroup $fuse:side
SELECT face 6
    ATTRIBUTE capsGroup $fuse:side
SELECT body
```

- Properties assigned to capsGroups

session02/07\_masstran\_f118.py

---

```
# Define material properties
unobtainium = {"density" : 7850}
madeupium   = {"density" : 6890}

# Set the materials
masstran.input.Material = {"Unobtainium": unobtainium,
                           "Madeupium" : madeupium}

# Define shell properties
shell_1 = {"propertyType" : "Shell",
           "material"      : "unobtainium",
           "membraneThickness" : 0.2}

shell_2 = {"propertyType" : "Shell",
           "material"      : "madeupium",
           "membraneThickness" : 0.3}

# Associate the shell property with capsGroups defined on the geometry
masstran.input.Property = {"wing:faces" : shell_1, "htail:faces": shell_1,
                           "fuse:nose"   : shell_1, "fuse:tail"  : shell_1,
                           "vtail:faces" : shell_2, "fuse:side"   : shell_2}
```

---

## Thickness

- Plot airfoil polars for a range of airfoil thicknesses
  - Start from a copy of session02/04\_xfoil\_Camber\_Plot.py

## Custom masstran analysis

- Load wing, htail and fuselage into a masstranAIM
  - Start from a copy of session02/05\_masstran\_f118\_AIM.py

## New Shells and Material

- Change `capsGroup` value for the top and bottom faces of the fuselage for F-118C (either the same or two different values)
- Starting with session02/07\_masstran\_f118.py, create a new shell and/or material for the newly created `capsGroup(s)`

## F-118C CG Location

- Using session02/07\_masstran\_f118.py, create an array of F-118C CG x-locations by modifying the wing:xroot location