

# Computational Aircraft Prototype Syntheses



## Training Session 4.1 Aero Modeling: AVL and masstran

**Marshall Galbraith**

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

Massachusetts Institute of Technology

**Bob Haines**

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

**John F. Dannenhoffer, III**

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

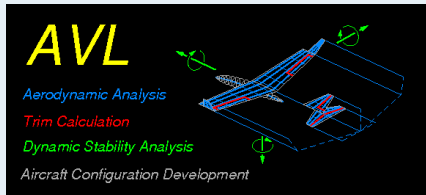
Syracuse University

- AVL Overview
  - AVL Geometry Definition
  - Reference Quantities
- Control Surfaces and Stability Derivatives
- AVL Eigenmode Analysis
  - Pure AVL
  - AVL and masstran
- Suggested Exercises

- Aerodynamic and flight-dynamic analysis of rigid aircraft

## Extended Vortex-Lattice Model

- Aerodynamic Components
  - Lifting surfaces
  - Slender bodies
- Control deflections
  - Via normal-vector tilting
  - Leading edge or trailing edge flaps
- General freestream description
  - $\alpha, \beta$  flow angles
  - $p, q, r$  aircraft rotation
- Aerodynamic outputs
  - forces and moments, in body or stability axes
  - Force and moment derivatives w.r.t. angles, rotations, controls



## Trim Calculation

- Operating variables
  - $\alpha, \beta$
  - $p, q, r$
  - control deflections
- Constraints
  - direct constraints on variables
  - indirect constraints via specified CL, moments
  - level or banked horizontal flight
  - steady pitch rate (looping) flight

## Eigenmode analysis

- Predicts flight stability characteristics
- Rigid-body, quasi-steady aero model
- Eigenvalue root progression with a parameter
- Display of eigenmode motion in real time

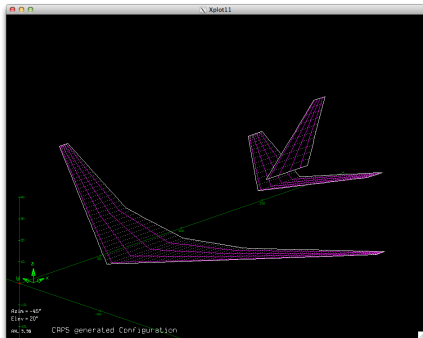
## Geometry specified with airfoil sections

```
#-----
SURFACE
WING
#Nchordwise  Cspace  Nspanwise  Sspace
1            1.0     16          -2.0
YDUPLICATE
0.0

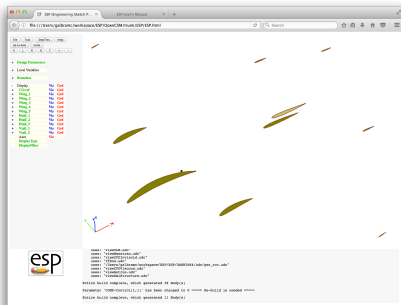
SECTION
#Xle  Yle  Zle  Chord  Ainc  Nspanwise  Sspace
-0.25  0.  0.  1.000  0.  8          1.0
AIRFOIL
naca2412.dat

SECTION
#Xle  Yle  Zle  Chord  Ainc  Nspanwise  Sspace
-0.175 7.5  0.5  0.700  0.  0          0
AIRFOIL
naca0012.dat
```

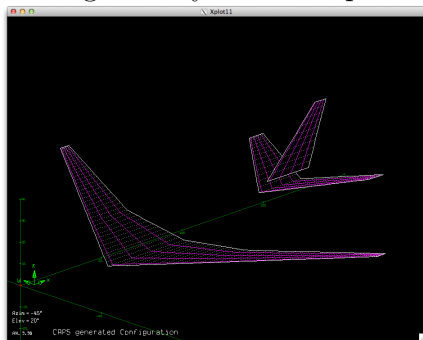
## VLM geometry with flat panels



## Geometry airfoil sections with ESP



## VLM geometry with flat panels



# Geometry airfoil sections with ESP



View Concept and VLM in ESP

```
cd $ESP_ROOT/training/ESP
serveCSM transport.csm
```

VIEW:Concept	1
VIEW:VLM	1

session4.1/avl\_1\_TransportGeom.py

---

```
# Load geometry [.csm] file
filename = os.path.join(".", "ESP", "transport.csm")
print ("\n==> Loading geometry from file \""+filename+"\"")
transport = myProblem.loadCAPS(filename)

# Change to VLM view
transport.setGeometryVal("VIEW:Concept", 0)
transport.setGeometryVal("VIEW:VLM", 1)

# view the geometry with the capsViewer
print ("\n==> Viewing transport bodies...")
transport.viewGeometry()

# Load AVL AIM
print ("\n==> Loading AVL aim...")
avl = myProblem.loadAIM(aim = "avlAIM",
                        analysisDir = "workDir_avl_1_TransportGeom")

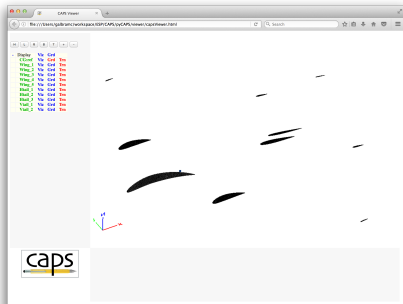
# view avl bodies with the capsViewer
print ("\n==> Viewing avl bodies...")
avl.viewGeometry()

# Close CAPS
myProblem.closeCAPS()
```

---



```
transport.viewGeometry()
```



avl.viewGeometry()



## ESP/viewVLM.udc

```
RESTORE  Vtail0m1
INTERSECT
  #ATTRIBUTE capsAIM      vlmAIMs
  ATTRIBUTE capsGroup     $Vtail
```

- Very rich input data set
  - Many geometric parameter
  - Multiple bodies
  - Many attributes on BODY/FACE/EDGE/NODE
- Not all error checking can be automated
- Significant user responsibility to check consistency
- Always check initial setup as much as possible

## AVL AIM Documentation

## AVL Input Header

!Sref	Cref	Bref
12.0	1.0	15.0
!Xref	Yref	Zref
0.0	0.0	0.0

## ESP/viewVLM.udc

```

RESTORE Wing0ml
INTERSECT
  ATTRIBUTE capsAIM vlmAIMs
  ATTRIBUTE capsReferenceArea wing:area
  ATTRIBUTE capsReferenceSpan wing:span
  ATTRIBUTE capsReferenceChord wing:mac
  ATTRIBUTE capsReferenceX wing:xroot+wing:mac/4

```

## capsReference\* attributes

Sref	capsReferenceArea	area for coefficients ( $C_L$ , $C_D$ , $C_m$ , etc)
Cref	capsReferenceChord	chord for pitching moment ( $C_m$ )
Bref	capsReferenceSpan	span for roll,yaw moments ( $C_l$ , $C_n$ )
Xref	capsReferenceX	location for moments, rotation rates

capsReference\* attributes on one or more bodies (consistent)

- AVL Overview
  - AVL Geometry Definition
  - Reference Quantities
- Control Surfaces and Stability Derivatives
- AVL Eigenmode Analysis
  - Pure AVL
  - AVL and masstran
- Suggested Exercises

- Controls specified with airfoil sections
- Airfoil interpolation?

---

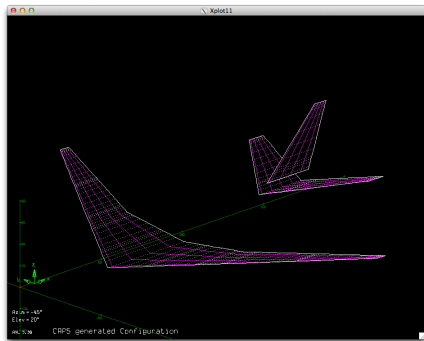
```
#-----
SURFACE
STAB
#Nchordwise  Cspace  Nspanwise  Sspace
1            1.0     7           -2.0
YDUPLICATE
0.0

SECTION
#Xle  Yle  Zle  Chord  Ainc  Nspanwise  Sspace
6.0   0.   0.0  0.4    0.    7           -1.25
CONTROL
elevator  1.0  0.0  0.  0.  0.  1

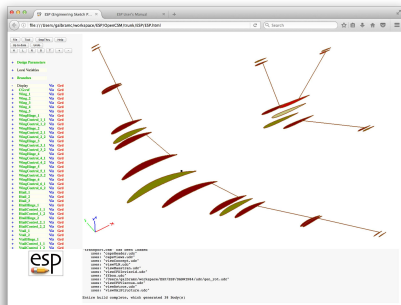
SECTION
#Xle  Yle  Zle  Chord  Ainc  Nspanwise  Sspace
-0.075 2.00  0.0  0.3    0.    0           0
CONTROL
elevator  1.0  0.0  0.  0.  0.  1
```

---

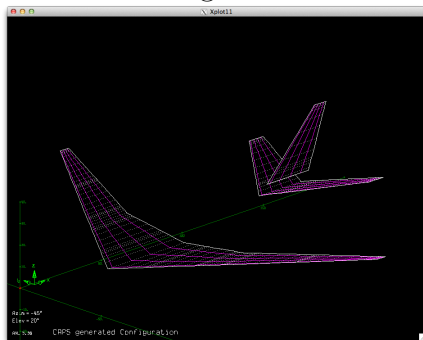
Mesh clustering around controls



Controls specified with airfoil sections

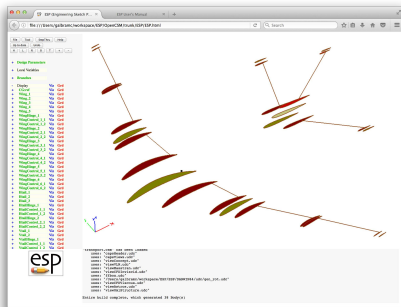


Mesh clustering around controls





## Controls specified with airfoil sections



## View Concept and VLM in ESP

```
cd $ESP_ROOT/training/ESP
serveCSM transport.csm
```

VIEW:Concept 1

VIEW:VLM 1

COMP:Control 1

- vlmControl\_“Name” specifies a section with a control surface
  - “Name” is the name of the control surface
  - Value is chord fraction of hinge line

### session4.1/avlPlaneVanilla.csm

```
UDPRIM      naca Thickness wing:thick Camber wing:camber
SCALE       wing:croot
ROTATEX     90      0      0
TRANSLATE   wing:xroot 0      wing:zroot
ATTRIBUTE   capsGroup      $Wing
ATTRIBUTE   vlmControl_AileronLeft 0.8 #Hinge line 80% chord
ATTRIBUTE   vlmControl_AileronRight 0.8 #Hinge line 80% chord
```

### session4.1/avl\_2\_PlaneVanillaControl.py

```
# Set control surface parameters
aileronLeft  = {"deflectionAngle" : -25.0}
aileronRight = {"deflectionAngle" : 25.0}
elevator     = {"deflectionAngle" : 5.0}
rudder       = {"deflectionAngle" : -2.0}

avl.setAnalysisVal("AVL_Control", [ ("AileronLeft" , aileronLeft ),
                                     ("AileronRight", aileronRight),
                                     ("Elevator"    , elevator    ),
                                     ("Rudder"      , rudder       )])
```

## ESP/transport.csm

---

```
# wing hinge lines
DIMENSION wing:hinge      6 9 1 # ymin          ymax
#                          theta  x/c   y/span  z/t      x/c   y/span  z/t      gap  grp
DESPMTR   wing:hinge      "-10.0;  0.75; -0.98;  0.50;  0.75; -0.70; 0.50;  0.25; 1;  \ left aileron
+10.0;  0.75; -0.69;  0.00;  0.75; -0.43; 0.00;  0.25; 2;  \ left oflap
+15.0;  0.85; -0.33;  0.00;  0.90; -0.14; 0.00;  0.25; 3;  \ left iflap
+15.0;  0.90;  0.14;  0.00;  0.85;  0.33; 0.00;  0.25; 3;  \ rite iflap
+10.0;  0.75;  0.43;  0.00;  0.75;  0.69; 0.00;  0.25; 2;  \ rite oflap
+10.0;  0.75;  0.70;  0.50;  0.75;  0.98; 0.50;  0.25; 4" # rite aileron
```

---

## ESP/viewVLM.udc

---

```
ATTRIBUTE capsGroup          $Wing
ATTRIBUTE capsDiscipline     $Aerodynamic
ATTRIBUTE _name              !$WingControl_+ihinge+$_1
ATTRIBUTE !$vlmControl_WingControl+tagIndex xoverc1
```

---

## session4.1/avl\_3\_TransportControl.py

---

```
# Set up control surface deflections based on the information in the csm file
controls = []
```

```
hinge = transport.getGeometryVal("wing:hinge")
for i in range(len(hinge)):
    controls.append(("WingControl_"+str(int(hinge[i][8])), {"deflectionAngle": hinge[i][0]}))
```

---

- AVL Overview
  - AVL Geometry Definition
  - Reference Quantities
- Control Surfaces and Stability Derivatives
- AVL Eigenmode Analysis
  - Pure AVL
  - AVL and masstran
- Suggested Exercises

## Eigenmode analysis

- Requires realistic:
  - Configuration
  - Mass, CG, and inertia data
  - Flight conditions
- All in with dimensional units
  - Units of body defined by `capsLength` attribute

ESP/transport.csm

---

```
# Define length units of the geometry
ATTRIBUTE capsLength      $ft
```

---

- Stable configuration: all negative real Eigen values

- Specifying Eigenmode Analysis dimensional inputs

session4.1/avl\_4\_TransportEigen.py

```
I = "massInertia"
# Inspired by the b737.mass avl example file
#
#           mass           CGx  CGy  CGz           Ixx           Iyy           Izz
cockpit   = {"mass": [ 3000, "lb"], "CG": [[ 8, 0, 5], "ft"], I: [[0. , 0. , 0. ], "lb*ft^2"]}
wing      = {"mass": [19420, "lb"], "CG": [[ 78, 0, -1], "ft"], I: [[8.0e6, 0.1e6, 8.1e6], "lb*ft^2"]}
fuselage  = {"mass": [33720, "lb"], "CG": [[105, 0, 2], "ft"], I: [[0.7e6, 18.9e6, 19.6e6], "lb*ft^2"]}
tailcone  = {"mass": [ 310, "lb"], "CG": [[145, 0, 0], "ft"], I: [[0. , 0. , 0. ], "lb*ft^2"]}
Htail     = {"mass": [ 528, "lb"], "CG": [[160, 0, 2], "ft"], I: [[0.0e6, 0.0e6, 0.0e6], "lb*ft^2"]}
Vtail     = {"mass": [ 616, "lb"], "CG": [[100, 0, 8], "ft"], I: [[0.1e6, 0.0e6, 0.1e6], "lb*ft^2"]}
Main_gear = {"mass": [ 4500, "lb"], "CG": [[ 76, 0, -4], "ft"], I: [[0.5e6, 0.0 , 0.5e6], "lb*ft^2"]}
Nose_gear = {"mass": [ 1250, "lb"], "CG": [[ 36, 0, -5], "ft"], I: [[0. , 0. , 0. ], "lb*ft^2"]}

avl.setAnalysisVal("MassProp", [{"cockpit" , cockpit },
                                ("wing"      , wing      ),
                                ("fuselage"   , fuselage  ),
                                ("tailcone"   , tailcone  ),
                                ("Htail"      , Htail     ),
                                ("Vtail"      , Vtail     ),
                                ("Main_gear"  , Main_gear),
                                ("Nose_gear"  , Nose_gear)])

avl.setAnalysisVal("Gravity" , 32.18 , units="ft/s^2" )
avl.setAnalysisVal("Density" , 0.002378, units="slug/ft^3")
avl.setAnalysisVal("Velocity", 250.0 , units="m/s" )
```

- Check bodies passed to avl and masstran

## session4.1/avl\_masstran\_5\_Geom.py

```
# Change to VLM view and OmlStructure
transport.setGeometryVal("VIEW:Concept",    0)
transport.setGeometryVal("VIEW:VLM",        1)
transport.setGeometryVal("VIEW:OmlStructure", 1)

# Enable fuselage and lifting surfaces
transport.setGeometryVal("COMP:Wing"      , 1)
transport.setGeometryVal("COMP:Fuse"      , 1)
transport.setGeometryVal("COMP:Htail"     , 1)
transport.setGeometryVal("COMP:Vtail"     , 1)
transport.setGeometryVal("COMP:Control"   , 0)

avl = myProblem.loadAIM(aim = "avlAIM",
                        analysisDir = "workDir_avl_5_Geom")

print ("AVL geometry")
avl.viewGeometry()

masstran = myProblem.loadAIM(aim = "masstranAIM",
                             analysisDir = "workDir_masstran_5_Geom")

print ("Masstran geometry")
masstran.viewGeometry()
```

- Get mass properties from masstran

session4.1/avl\_masstran\_6\_Eigen.py

---

```
# Set materials
unobtainium = { "density" : 200 } # lb/ft^3
masstran.setAnalysisVal("Material", ("Unobtainium", unobtainium))

# Set property
shell = {"propertyType"      : "Shell",
        "material"          : "Unobtainium",
        "membraneThickness" : 0.02} # ft

masstran.setAnalysisVal("Property", [( "fuseSkin", shell),
                                     ( "wingSkin", shell),
                                     ( "htailSkin", shell),
                                     ( "vtailSkin", shell)])

print ("\n=> Computing mass properties...")
masstran.preAnalysis()
masstran.postAnalysis()

aircraft_mass = masstran.getAnalysisOutVal("Mass")
aircraft_CG   = masstran.getAnalysisOutVal("CG")
aircraft_I    = masstran.getAnalysisOutVal("I_Vector")

aircraft_skin = {"mass": [aircraft_mass, "lb"], "CG": [aircraft_CG, "ft"], "massInertia": [aircraft_I, "lb*ft^2"]}
```

---



- Pass mass properties to AVL

## session4.1/avl\_masstran\_6\_Eigen.py

```
# Taken from the b737.mass avl example file
I = "massInertia"
#
#           mass           CGx  CGy  CGz           Ixx  Iyy  Izz
cockpit   = {"mass":[3000, "lb"], "CG":[[ 8, 0., 5], "ft"], I:[[0.0 , 0.0, 0.0 ], "lb*ft^2"]}
Main_gear = {"mass":[4500, "lb"], "CG":[[ 86, 0., -4], "ft"], I:[[0.5e6, 0.0, 0.5e6], "lb*ft^2"]}
Nose_gear = {"mass":[1250, "lb"], "CG":[[ 26, 0., -5], "ft"], I:[[0.0 , 0.0, 0.0 ], "lb*ft^2"]}

avl.setAnalysisVal("MassProp", [("aircraft_skin", aircraft_skin),
                                ("cockpit"      , cockpit      ),
                                ("Main_gear"    , Main_gear    ),
                                ("Nose_gear"    , Nose_gear    )])

avl.setAnalysisVal("Gravity" , 32.18 , units="ft/s^2")
avl.setAnalysisVal("Density" , 0.002378, units="slug/ft^3")
avl.setAnalysisVal("Velocity", 250.0 , units="m/s")
```

## Multiple Shells

- Create multiple materials and shell properties for components

## Multiple AIMs

- Create multiple masstran AIM instances for components
  - `viewOmlStructure` has same `capsIntent` as F-118

## Main Gear

- Use `wing:xroot` and `wing:mac` to set main gear CGx

## Stable Transport

- Resize tail and modify mass properties to stabilize transport
  - See `session4.1/transport_Htail.py`
- Create your own