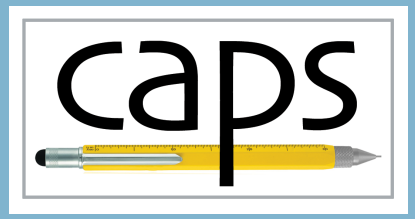
Computational Aircraft Prototype Syntheses



Training Session 1
CAPS Geometry

ESP v1.26

Marshall Galbraith
galbramc@mit.edu

Bob Haimes
haimes@mit.edu

Massachusetts Institute of Technology

John F. Dannenhoffer, III
jfdannen@syr.edu

Syracuse University

Overview

- Loading and viewing geometry via pyCAPS
  - pyCAPS.Problem
- Accessing/modifying DESPMTR
  - geometry.despmtr
  - geometry.save, .writeParameters, .readParameters
  - Value Object Sequence .value Shortcut
- Accessing SET and @values using OUTPMTR
  - geometry.outpmtr
- Suggested Exercises

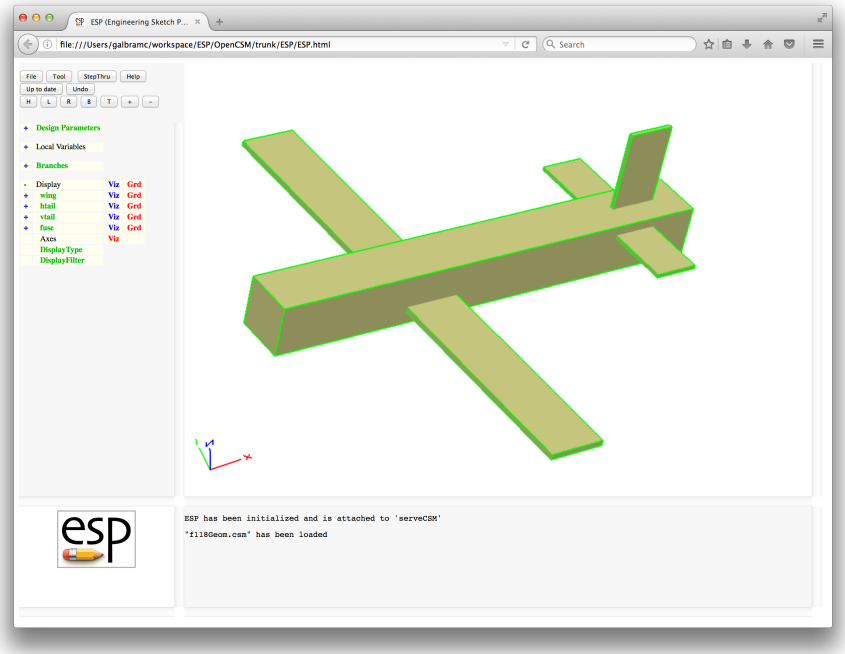
Overview

- Loading and viewing geometry via pyCAPS
  - pyCAPS.Problem
- Accessing/modifying DESPMTR
  - geometry.despmtr
  - geometry.save, .writeParameters, .readParameters
  - Value Object Sequence .value Shortcut
- Accessing SET and @values using OUTPMTR
  - geometry.outpmtr
- Suggested Exercises

F-118 Boxter Geometry

session01/f118-A.csm

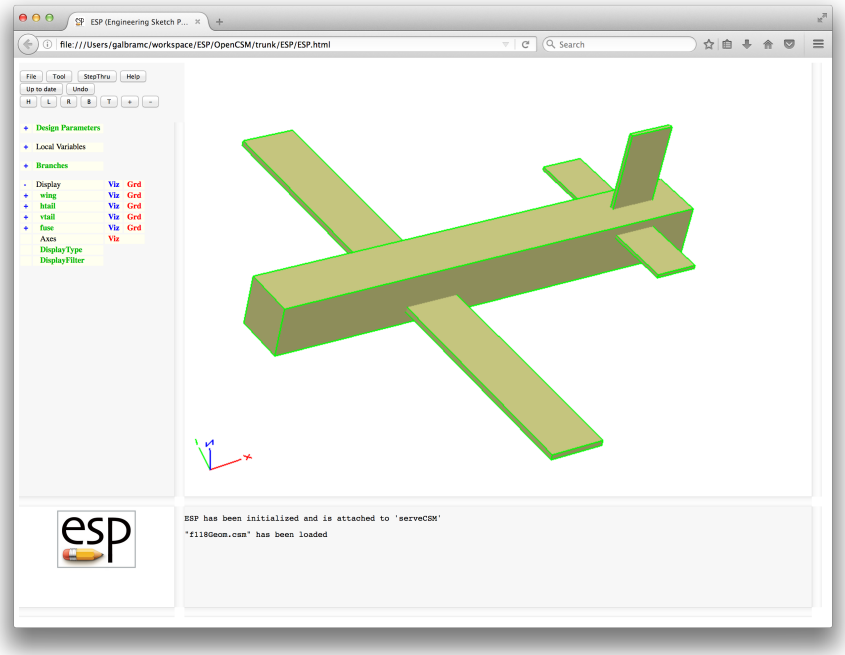
```
# F-118A Boxster

# wing design parameters
DESPMTR   wing:area         4240     # area
DESPMTR   wing:aspect       9.00     # aspect ratio
DESPMTR   wing:thick        0.10     # thickness ratio
DESPMTR   wing:xroot        54.0     # xloc at root LE
DESPMTR   wing:zroot        -5.0     # zloc at root LE

# horizontal tail design parameters
DESPMTR   htail:area        1210     # htail area
DESPMTR   htail:aspect      4.15     # htail aspect ratio
DESPMTR   htail:thick       0.08     # htail thickness
DESPMTR   htail:xroot        145     # xloc of root LE
DESPMTR   htail:zroot          5     # zloc of root LE

# vertical tail design parameters
CFGPMTR   nrow                 1
DIMENSION vtail nrow 6
DESPMTR   vtail             "610; \ # 1 vtail area
                           1.80; \ # 2 vtail aspect ratio
                           0.08; \ # 3 vtail thickness
                            150; \ # 4 xloc of root LE
                              0; \ # 5 yloc of root LE
                              9"   # 6 zloc of root LE

# fuselage design parameters
DESPMTR   fuse:length       180     # fuselage length
DESPMTR   fuse:width         20     # width  of fuselage
DESPMTR   fuse:height        20     # height of mid fuselage
```



F-118 Boxter Geometry

session01/f118-A.csm

```
#------------------------------------------------
# set available output parameters
OUTPMTR  wing:wet
OUTPMTR  wing:volume

DIMENSION htail:prop 2 1
OUTPMTR   htail:prop

DIMENSION vtail:prop 2 1
OUTPMTR   vtail:prop

OUTPMTR  fuse:wet
OUTPMTR  fuse:volume

#================================================
# Wing
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 _name   $Wing

SET  wing:wet    @area
SET  wing:volume @volume
```



pyCAPS Problem Object

- pyCAPS.Problem provides the context for a CAPS session
  - Multiple pyCAPS.Problems may be instantiated, but cannot interact
- `problemName` is a directory name for analysis/restart files
- `outLevel` sets verbosity (0-off, 1-default,2-debug)

session01/1_f118_Geom.py

```
#-------------------------------------------------------------------------------

# Import pyCAPS module
import pyCAPS

#-------------------------------------------------------------------------------

# Load geometry [.csm] file
# This parses the csm file, but does not build geometry
filename = "f118-A.csm"
print ('\n==> Loading geometry from file "'+filename+'"...')
capsProblem = pyCAPS.Problem(problemName = "design_f118",
                             capsFile = filename,
                             outLevel = 1)
```

Viewing Geometry with pyCAPS

- Problem.geometry is a pyCAPS.ProblemGeometry class instance
- Visualize with ESP using capsProblem.geometry.view
  - Geometry is built just-in-time

session01/1_f118_Geom.py

```
# capsProblem.geometry is a class representing bodies on the stack
# Build and view the geometry with ESP
print ('\n==> Bulding and viewing geometry...')
capsProblem.geometry.view()
```

Overview

- Loading and viewing geometry via pyCAPS
  - pyCAPS.Problem
- Accessing/modifying DESPMTR
  - geometry.despmtr
  - geometry.save, .writeParameters, .readParameters
  - Value Object Sequence .value Shortcut
- Accessing SET and @values using OUTPMTR
  - geometry.outpmtr
- Suggested Exercises

Modifying Geometry DESPMTR with pyCAPS

- DESPMTR ↔ geometry.despmtr Sequence
  - Sequence of ValueIn Objects
  - Keys correspond to DESPMTR names in csm file
  - Iterate despmtr.keys(), despmtr.values(), despmtr.items()
- CFGPMTR ↔ geometry.cfgpmtr Sequence
- CONPMTR ↔ geometry.conpmtr Sequence

session01/f118-A.csm

```
# fuselage design parameters
DESPMTR   fuse:length       180    # fuselage length
DESPMTR   fuse:width         20    # width  of fuselage
DESPMTR   fuse:height        20    # height of mid fuselage
```

session01/2_f118_DESPMTR.py

```
# Create an alias to geometry
f118 = capsProblem.geometry

# geometry.despmtr is a Sequence of ValueIn Objects of all DESPMTR defined in the csm file
# Set wide fuselage "fuse:width"
f118.despmtr["fuse:width"].value = 60
```

Modifying Geometry DESPMTR with pyCAPS

- DESPMTR ↔ geometry.despmtr Sequence

session01/f118-A.csm

```
# horizontal tail design parameters
DESPMTR   htail:area       1210    # htail area
DESPMTR   htail:aspect     4.15    # htail aspect ratio
DESPMTR   htail:thick      0.08    # htail thickness
DESPMTR   htail:xroot       145    # xloc of root LE
DESPMTR   htail:zroot         5    # zloc of root LE
```

session01/2_f118_DESPMTR.py

```
# Double the htail:area
htail_area = f118.despmtr["htail:area"].value
f118.despmtr["htail:area"].value = htail_area*2

print ("--> old htail:area = ", htail_area)
print ("--> new htail:area = ", f118.despmtr["htail:area"].value)
```

Modifying Geometry DESPMTR with pyCAPS

- Modifying DIMENSIONed DESPMTR
  - Modify copy of array values

session01/f118-A.csm

```
# vertical tail design parameters
CFGPMTR   nrow                1
DIMENSION vtail nrow 6
DESPMTR   vtail           "610; \ # 1 vtail area
                          1.80; \ # 2 vtail aspect ratio
                          0.08; \ # 3 vtail thickness
                           150; \ # 4 xloc of root LE
                             0; \ # 5 yloc of root LE
                             9"   # 6 zloc of root LE
```

session01/2_f118_DESPMTR.py

```
# Double the vtail area in an array
vtail = f118.despmtr["vtail"].value
vtail[0] *= 2
f118.despmtr["vtail"].value = vtail
```

Modifying Geometry DESPMTR with pyCAPS

- Swapping parameter values

session01/2_f118_DESPMTR.py

```
htail_area = f118.despmtr["htail:area"].value
wing_area  = f118.despmtr["wing:area"].value

# Swap "wing:area" and "htail:area"
f118.despmtr["htail:area"].value = wing_area
f118.despmtr["wing:area"].value  = htail_area/2
```

Modifying Geometry DESPMTR with pyCAPS

- Create a 2nd vertical tail

session01/f118-A.csm

```
CFGPMTR   nrow                1
DIMENSION vtail nrow 6
DESPMTR   vtail            "610; \ # 1 vtail area
                           1.80; \ # 2 vtail aspect ratio
                           0.08; \ # 3 vtail thickness
                            150; \ # 4 xloc of root LE
                              0; \ # 5 yloc of root LE
                              9"   # 6 zloc of root LE
```

```
PATBEG i vtail.nrow
    SET       vtail:span      sqrt(vtail[i,1]*vtail[i,2])
    SET       vtail:chord     vtail[i,1]/vtail:span
    SET       vtail:thick     vtail:chord*vtail[i,3]

    BOX  vtail[i,4]  vtail[i,5]-vtail:thick/2  vtail[i,6]   vtail:chord  vtail:thick  vtail:span
```

session01/2_f118_DESPMTR.py

```
# Make a 2nd vertical tail
vtail = [vtail[:],vtail[:]]                              # create a matrix
f118.cfgpmtr["nrow"].value = 2                           # change the CSM matrix size
vtail[0][4] = -f118.despmtr["fuse:width"].value/2*0.8  # spanwise tail positions
vtail[1][4] =  f118.despmtr["fuse:width"].value/2*0.8
f118.despmtr["vtail"].value = vtail
```

Modifying Geometry DESPMTR with pyCAPS

- Display all design parameters

session01/2_f118_DESPMTR.py

```
# Display all design parameters
print ('\n==> All design parameters')
for key in f118.despmtr.keys():
    print ('--> {:<12}'.format(key), '=', f118.despmtr[key].value)
```

Saving Static Geometry with pyCAPS

- Modified geometry is saved with geometry.save
  - Available extensions: .egads .stp .step .igs .iges .brep

session01/2_f118_DESPMTR.py

```
# Build and view the geometry with ESP
print ('\n==> Bulding and viewing Wide Body geometry...')
f118.view()
```

session01/3_f118_Save.py

```
# Build and save geometry
print ('\n==> Bulding and saving Wide Body geometry...')
f118.save("3_f118_Save_Wide.egads")
```

- View geometry with:
  - serveESP 3_f118_Save_Wide.egads
  - serveESP 3_f118_Save_Canard.egads

The Value Object Sequence .value shortcut

- Value Objects have several methods
- 90% of usage is modifying .value property
- Sequence of Value Objects provides shortcut to .value property

session01/f118-A.csm

```
# vertical tail design parameters
CFGPMTR   nrow                1
DIMENSION vtail nrow 6
DESPMTR   vtail           "610; \ # 1 vtail area
                          1.80; \ # 2 vtail aspect ratio
                          0.08; \ # 3 vtail thickness
                           150; \ # 4 xloc of root LE
                             0; \ # 5 yloc of root LE
                             9"   # 6 zloc of root LE
```

session01/2_f118_DESPMTR.py

```
# Double the vtail area in an array
vtail = f118.despmtr["vtail"].value
vtail[0] *= 2
f118.despmtr["vtail"].value = vtail
```

session01/4_f118_Shortcut.py

```
# Double the "vtail" area in an array
vtail = f118.despmtr.vtail
vtail[0] *= 2
f118.despmtr.vtail = vtail
```

The Value Object Sequence .value shortcut

- Value Objects have several methods
- 90% of usage is modifying .value property
- Colons in OpenCSM names create "groups" in ESP UI
- pyCAPS Value Object Sequence is "grouped"

session01/f118-A.csm

```
# fuselage design parameters
DESPMTR   fuse:length       180    # fuselage length
DESPMTR   fuse:width         20    # width  of fuselage
DESPMTR   fuse:height        20    # height of mid fuselage
```

session01/4_f118_Shortcut.py

```
# Set wide fuselage "fuse:width" via shortcuts
f118.despmtr["fuse:width"].value = 60
f118.despmtr.fuse.width = 60
f118.despmtr["fuse"].width = 60
f118.despmtr["fuse"]["width"].value = 60

# Double the "htail:area"
htail_area = f118.despmtr["htail"].area
f118.despmtr["htail"].area = htail_area*2
```

```
# Display grouped design parameters
print ('\n==> Grouped parameters')
for group in ["fuse", "wing", "htail"]:
    print ('-->', group)
    for key in f118.despmtr[group].keys():
        print ('    {:<12}'.format(group+":"+key),
               '=', f118.despmtr[group][key].value)
```

Saving Parameters with pyCAPS

- Geometry parameters are saved with geometry.writeParameters and geometry.readParameters

session01/5_f118_writeParameters.py

```
f118.despmtr["htail:area"].value = htail_area*2

# Double the vtail area in an array
vtail = f118.despmtr["vtail"].value
vtail[0] *= 2
f118.despmtr["vtail"].value = vtail

# Write geometry parameters
paramFile = "f118_4.params"
print ('\n==> Saving parameters to "'+paramFile+'"...')
f118.writeParameters(paramFile)

print ('\n==> Loading 2nd geometry from file "'+filename+'"...')
capsProblem2 = pyCAPS.Problem(problemName = "design_f118",
                              capsFile = filename,
                              phaseName = "4_writeParameters_2",
                              outLevel = 0)

# Read back in the parameters
print ('\n==> Reading parameters from "'+paramFile+'"...')
capsProblem2.geometry.readParameters(paramFile)
```

Overview

- Loading and viewing geometry via pyCAPS
  - pyCAPS.Problem
- Accessing/modifying DESPMTR
  - geometry.despmtr
  - geometry.save, .writeParameters, .readParameters
  - Value Object Sequence .value Shortcut
- Accessing SET and @values using OUTPMTR
  - geometry.outpmtr
- Suggested Exercises

Accessing Geometry OUTPMTR with pyCAPS

- OUTPMTR accessed with geometry.outpmtr Sequence of ValueOut Objects

session01/f118-A.csm

```
#-----------------------------------------------
# set available output parameters
OUTPMTR  wing:wet
OUTPMTR  wing:volume
```

```
BOX  wing:xroot  -wing:span/2  wing:zroot   wing:chord  wing:span  wing:chord*wing:thick
SELECT body
    ATTRIBUTE _name   $Wing

SET  wing:wet    @area
SET  wing:volume @volume
```

session01/6_f118_OUTPMTR.py

```
# Create an alias to geometry
f118 = capsProblem.geometry

# Build and print OUTPMTRs
print ("--> wing:wet      =", f118.outpmtr["wing:wet"].value )
print ("--> wing:volume   =", f118.outpmtr.wing.volume )
```

Accessing Geometry OUTPMTR with pyCAPS

- OUTPMTR can be DIMENSIONed array/matrix

session01/f118-A.csm

```
DIMENSION htail:prop 2 1
OUTPMTR   htail:prop
```

```
BOX  htail:xroot  -htail:span/2  htail:zroot   htail:chord  htail:span  htail:chord*htail:thick
SELECT body
    ATTRIBUTE _name   $Htail

SET  htail:prop[1] @area
SET  htail:prop[2] @volume
```

session01/6_f118_OUTPMTR.py

```
# Accessing DIMENSIONed OUTPMTR
htail_prop = f118.outpmtr["htail"].prop
print ("--> htail:prop[0] =", htail_prop[0] )
print ("--> htail:prop[1] =", htail_prop[1] )
```

Accessing Geometry OUTPMTR with pyCAPS

- None returned for any OUTPMTR not SET

session01/f118-A.csm

```
DIMENSION vtail:prop 2 1
OUTPMTR   vtail:prop

OUTPMTR  fuse:wet
OUTPMTR  fuse:volume
```

```
    SET  vtail:prop[1] vtail:prop[1]+@area
    # vtail:prop[2] is not set
```

```
# fuse:wet and fuse:volume not set
```

session01/6_f118_OUTPMTR.py

```
# Accessing OUTPMTR that has not been set
print ("--> vtail:prop    =", f118.outpmtr["vtail:prop"].value )
print ()
print ("--> fuse:wet      =", f118.outpmtr["fuse:wet"].value )
print ("--> fuse:volume   =", f118.outpmtr.fuse.volume )
```

Accessing Geometry OUTPMTR with pyCAPS

- Accessing non-OUTPMTR raises KeyError/AttributeError

session01/f118-A.csm

```
#-------------------------------------------------
# set available output parameters
OUTPMTR  wing:wet
OUTPMTR  wing:volume

DIMENSION htail:prop 2 1
OUTPMTR   htail:prop

DIMENSION vtail:prop 2 1
OUTPMTR   vtail:prop

OUTPMTR  fuse:wet
OUTPMTR  fuse:volume

#=================================================
# Wing
SET       wing:span      sqrt(wing:aspect*wing:area)
```

session01/6_f118_OUTPMTR.py

```
# Attempt to get a SET value not defined as OUTPMTR (raises KeyError/AttributeError)
print ("--> wing:span     =", f118.outpmtr.wing.span )
```

Suggested Exercises

Fix f118-A.csm

- `SET` fuse:wet and fuse:volume in session01/f118-A.csm
- Add wing:span as OUTPMTR in session01/f118-A.csm
- Rerun session01/6_f118_OUTPMTR.py

Custom f118-A.csm

- Customize the f118-A.csm with geometry.despmtr
  - Start from a copy of session01/2_f118_DESPMTR.py