

# Engineering Sketch Pad (ESP)



## Advanced User Training Session 05 Building Large Models

**John F. Dannenhoffer, III**

[john@geocentrictech.com](mailto:john@geocentrictech.com)

Geocentric Technologies LLC

updated for v1.29+

- Background
- Directory structure
- Making a component
  - wing engineering parameters
  - interface to `liftsurf0ml`
  - wing component parameters
  - making the OML
  - other views
- `writeFile` UDC
- Homework exercises

- During the design of an aircraft, various coupled models are needed
  - different disciplines
    - structures
    - controls
    - aerodynamics
    - ...
  - different fidelities
    - conceptual design
    - preliminary design
    - detailed design
- There needs to be communication between these models

- **One** definition for each component:
  - wing — inboard and outboard sections (with controls)
  - fuselage — tube with cockpit and raked tail
  - horizontal tail — one section (with controls)
  - vertical tail — one section (with controls)
  - nacelle — flow-through
  - pylon — one section
  - payloads — cockpit, passenger compartments, galleys, baggage hold, fuel tank

- One of the strengths of ESP is to be able to have multiple “views” of a single configuration
  - tailored to a specific analysis method
  - driven by a single set of Design Parameters
  - attributed so that “common” features could be linked together
  - commensurate with the meshing requirements of the analysis
- Biggest problem is that such models can get very large
  - break up into nested user-defined components (UDCs)

- `aircraft` — directory for each aircraft project
  - `b787`
  - `drone`
  - ...
- `components` — directory for standard components
  - `tubebody` — useful for fuselage, ...
  - `liftSurf` — useful for wing, tails, ...

- Define the engineering CFGPMTRs and DESPMTRs
  - area, aspect ratio, taper ratio, ...
- Translate into component parameters
  - leading-edge locations, trailing-edge locations, ...
- Call the appropriate UDC



# Defining the Wing Engineering Parameters

Excerpted from b787/wingPmtrs.udc

INTERFACE . ALL

PREFIX wing

# wing planform design parameters

|         |           |      |                            |
|---------|-----------|------|----------------------------|
| DESPMTR | :area     | 4240 | # area                     |
| DESPMTR | :aspect   | 9.00 | # aspect ratio             |
| DESPMTR | :taperi   | 0.48 | # inboard taper ratio      |
| DESPMTR | :tapero   | 0.23 | # outboard taper ratio     |
| DESPMTR | :sweep    | 35.0 | # leading edge sweep       |
| DESPMTR | :dihedral | 7.0  | # dihedral                 |
| DESPMTR | :break    | 0.37 | # inboard/outboard         |
| DESPMTR | :alphar   | -1.0 | # setting angle at root    |
| DESPMTR | :thickr   | 0.10 | # thickness ratio at root  |
| DESPMTR | :camberr  | 0.08 | # camber ratio at root     |
| DESPMTR | :alphab   | -3.0 | # setting angle at break   |
| DESPMTR | :thickb   | 0.15 | # thickness ratio at break |
| DESPMTR | :camberb  | 0.04 | # camber ratio at break    |
| DESPMTR | :alphat   | -8.0 | # setting angle at tip     |
| DESPMTR | :thickt   | 0.08 | # thickness ratio at tip   |
| DESPMTR | :cambert  | 0.01 | # camber ratio at tip      |
| DESPMTR | :xroot    | 50.0 | # xloc at root LE          |
| DESPMTR | :zroot    | -8.0 | # zloc at root LE          |

# Interface to liftsurfOml

```
# makes a lifting-surface SolidBody, using the following parameters:
#   :filename          name of .csm file for making airfoil
#   :pmtrname          list of parameter name (without :) passed to :filename
#   :nsect             number of Xsects
#   :xle[:nsect]       x-coordinates of leading edge
#   :yle[:nsect]       y-coordinates of leading edge
#   :zle[:nsect]       z-coordinates of leading edge
#   :xte[:nsect]       x-coordinates of trailing edge
#   :yte[:nsect]       y-coordinates of trailing edge
#   :zte[:nsect]       z-coordinates of trailing edge
#   :tiptreat          (optional) tip treatment factor
#
#   If prefix():oml already exists
#       returns immediately
#   Else
#       SolidBody (without controls) is generated and STOREd as prefix():oml
#       all Edges and Faces get a tagComp=prefix() attribute
#       leading Edge(s) get a tagType=leading attribute
#       trailing Edge(s) get a tagType=trailing attribute
```



# Translating to Component Parameters (1)

Excerpted from b787.csm

```
# local variables
```

```
SET      :span      sqrt(:aspect*:area)
```

```
SET      :yroot     0
```

```
SET      :ytip      :span/2
```

```
SET      :xtip      :xroot+:ytip*tand(:sweep)
```

```
SET      :ztip      :zroot+:ytip*tand(:dihedral)
```

```
SET      :ybreak    :ytip*:break
```

```
SET      :xbreak    :xroot+:ybreak*tand(:sweep)
```

```
SET      :zbreak    :zroot+:ybreak*tand(:dihedral)
```

```
SET      :chordr    :area/((:yroot+:ybreak)*(:taperi+1)+(:ytip-:ybreak)*:taperi*(:
```

```
SET      :chordb    :chordr*:taperi
```

```
SET      :chordt    :chordb*:tapero
```

```
SET      :mac       sqrt(:area/:aspect)
```



# Translating to Component Parameters (2)

Excerpted from b787.csm

```
# variables for liftsurf0ml
CFGPMTR   :nsect      3
DIMENSION :xle        1 :nsect
DIMENSION :yle        1 :nsect
DIMENSION :zle        1 :nsect
DIMENSION :xte        1 :nsect
DIMENSION :yte        1 :nsect
DIMENSION :zte        1 :nsect

DIMENSION :type       1 :nsect
DIMENSION :thick      1 :nsect
DIMENSION :camber     1 :nsect
```

```

SET      :xle      ":xroot;\
                  :xbreak;\
                  :xtip"
SET      :yle      "0;\
                  :ybreak;\
                  :ytip"
SET      :zle      ":zroot;\
                  :zbreak;\
                  :ztip"
SET      :xte      ":xroot+:chordr*cosd(:alphan);\
                  :xbreak+:chordb*cosd(:alphan);\
                  :xtip+:chordt*cosd(:alphan)"
SET      :yte      "0;\
                  :ybreak;\
                  :ytip"
SET      :zte      ":zroot+:chordr*sind(:alphan);\
                  :zbreak+:chordb*sind(:alphan);\
                  :ztip+:chordt*sind(:alphan)"

SET      :type      1
SET      :thick      ":thickr; :thickb; :thickt"
SET      :camber      ":camber; :camberb; :cambert"

```



# Making the Wing OML (1)

Excerpted from b787.csm

```
UDPRIM      $/../../components/liftsurfOml

# rename and retag wing:oml to make rwing:oml
RESTORE     wing:oml

SELECT      BODY
SELECT      FACE
ATTRIBUTE   tagComp    $rwing
SELECT      EDGE
ATTRIBUTE   tagComp    $rwing
STORE       rwing:oml
```



# Making the Wing OML (2)

Excerpted from b787.csm

```
# create lwing:oml by MIRRORing wing:oml and retagging
```

```
RESTORE    wing:oml
```

```
MIRROR     0  1  0
```

```
SELECT     BODY
```

```
SELECT     FACE
```

```
ATTRIBUTE  tagComp    $lwing
```

```
SELECT     EDGE
```

```
ATTRIBUTE  tagComp    $lwing
```

```
STORE      lwing:oml
```

```
# make lifting-surface SolidBody (with controls) from anoml Solid Body, using:
#   :hinge[ictrl,9]  hinge parameters
#   rname            name of rite-side oml Body
#   lname            name of left-side oml Body (or .)
#
# restores rname:oml
# add control surfaces (using prefix():hinge)
# stores   rname:ctrl
#
# restores lname:oml
# add control surfaces (using prefix():hinge)
# stores   lname:ctrl
```



# Interface to liftsurfAvl.udc

```
# makes lifting-surface SheetBodys from an oml SolidBody, using:
#   :hinge[ictrl,9]  hinge parameters
#   rname            name of rite-side oml Body
#   lname            name of left-side oml Body (or .)
#
#   restores rname:oml
#   create SheetBodys at breaks in definition
#   create SheetBodys at ends of control surfaces
#   stores group as rname:avl
#
#   restores lname:oml
#   create SheetBodys at breaks in definition
#   create SheetBodys at ends of control surfaces
#   stores group as lname:avl
```

```
# :spar2[*,1]      y inboard
#           2      >0 x inboard
#           -n
#           -99 make same as x outboard
#           3      y outboard
#           4      >0 x outboard
#           -n
#           -99 make same as x inboard
#
# :ribs[*,1]      sparA (source rib)
#           2      sparB (target rib)
#           3      npnt equally-spaced points on sparA
#           -1     1 point at beginning of sparA
#           -2     1 point at end of sparA
#           4      0 use equally-spaces points on sparB
#           1      1 make rib perpendicular to sparB
```



# Interface to tubebodyOml.udc

## Contents of b787/fusePmtrs.udc

```
# makes a tube-like SolidBody, using the following parameters:
#   :filename          name of .csm file for making Xsect
#   :pmtrname          list of paramater names (without :) passed to :filename
#   :nsect             number of Xsects
#   :x[:nsect]         x-locations
#   :roty[:nsect]      (optional) rotation about y
#   :conty[:nsect]     continuity (2=C2, -1=C1/rev, +1=C1/fwd, 0=C0)
#   :noselist[2]       (optional) nose y-radius; nose z-radius
#   :taillist[2]       (optional) tail y-radius; tail z-radius
#
#   If prefix():oml already exists
#       returns immediately
#   Else
#       SolidBody is generated and STOREd as prefix():oml
#       all Edges and Faces get a tagComp=prefix() attribute
```



# UDC for Writing a File (1)

## Example template file

```
# writeFile1.txt
# written by John Dannenhoffer

SET      three  3
SET      four   $four

SET      ONE    {one}
SET      TWO    {two}
SET      THREE  {three}
SET      FOUR   {four}
SET      TWELVE {one}-{two}

ASSERT   TWELVE 12

POINT    ONE TWO THREE

END
```



# UDC for Writing a File (2)

## Example of using a template file

```
# writeFile1
# written by John Dannenhoffer

SET      one    1
SET      two    2
SET      three  3
SET      four   $four

UDPARC   writeFile template  $$/writeFile1.txt
UDPARC   writeFile outfile   $writeFileNew1.csm
UDPRIM   writeFile subs      "!${one}+one\
                               +${two}2\
                               +${three}three\
                               +${four}+four"

# verify that good file got written
UDPRIM   csm          filename  $writeFileNew1.csm

END
```

## Example of using an inline template file

```
# writeFile2
# written by John Dannenhoffer

SET      one    1
SET      two    2
SET      three  3
SET      four   $four

UDPARG   writeFile template <<
# writeFile2.txt
# written by John Dannenhoffer

...

END
>>
UDPARG   writeFile outfile $writeFileNew2.csm
UDPRIM   writeFile subs   "!\${one}+one\
                           +${two}2\
                           +${three}three\
                           +${four}+four"
```

- Start `serveESP` `exercises/session05/aircraft/drone/plugs/fuselage` and execute **Tools**→**Plugs** using `exercises/session05/aircraft/drone/plugs/fuselage.cloud` as the cloud of points. You should **ExecutePhase1** to 1 pass and **ExecutePhase2** for (up to) 25 passes.
- Modify `exercises/session05/aircraft/drone/plugs/fuselage.csm` in the following ways:
  - add a `CFGPMTR` named `writeUdc` and initialize it to 0
  - at the end, if `writeUdc`  $\neq$  0, write code to create a file named `fuselagePmtrs.udc` and a file named `fuselageXsect.csm` (Hint: use the `writeFile` UDP)

- Re-start `serveESP exercises/session05/aircraft/drone/plugs/fuselage` and
  - re-execute **Plugs** (as before)
  - change `writeUdc` to 1
  - **PressToRe-build**
- Modify `exercises/session05/aircraft/drone/drone.csm` to generate a fuselage in much the same way as the fuselage was added in `b787.csm`
  - add a UDPRIM statement to load the `fuselagePmtrs.udc` that you just created
  - add a UDPRIM statement to load the `tubebody0m1.udc`
- Execute `serveESP exercises/session05/aircraft/drone` to verify that you can make a fuselage

- Modify `exercises/session05/aircraft/drone.csm` in the following ways:
  - change `components` to include `lwing` and `rwing`
  - uncomment the section that makes `rwing:oml` and `lwing:oml`
- Re-run `serveESP` on `exercises/session05/aircraft/drone.csm`
- Add in `ltail:oml`, `rtail:oml`, and `vtail:oml` by editing `exercises/session05/aircraft/drone.csm` and re-executing

- Start `serveESP exercises/session05/aircraft/drone/package/nacelle` and execute **Tools**→**Miniopt** and **Minimize** for 100 iterations
- Modify `exercises/session05/aircraft/drone/packaging/nacelle.csm` in the following ways:
  - add a CFGPMTR named `writeUdc` and initialize it to 0
  - at the end if `writeUdc`  $\neq$  0, write code to create a file named `nacellePmtrs.udc` and a file named `nacelleXsect.csm`
- Bonus 1: add a straight duct rearward of the engine exit and add a blunt Face to connect with the nacelle at the rearward extent
- Bonus 2: add a straight duct forward of the engine inlet and connect it with the nacelle with a rounded lip

- Re-start `serveESP exercises/session05/aircraft/drone/packaging/nacelle`
  - and re-execute **Miniopt** (as before)
  - change `writeUdc` to 1
  - **PressToRe-build**
- Modify `exercises/session05/aircraft/drone.csm` to generate a nacelle (in much the same way as the fuselage was added above)
- Execute `serveESP exercises/session05/aircraft/drone` to verify that you can make nacelles

- Modify `exercises/session05/aircraft/drone.csm` in order to incorporate the pylons
- You should now have all the aircraft components flying in formation
- If you change `showPayloads` to 1 you can see all the payloads too (you may need to make some Faces transparent)
- IMPORTANT: It is a best-practice to build up a complex configuration component-by-component as was done here

- Following the process in `b787.csm` for adding Views:
  - add `VIEW = 1` to generate an outer-mold-line (OML) that is UNION of all the components
  - add `VIEW = 2` to add control surfaces to the single OML created above
  - add `VIEW = 3` to create the SheetBodyys that are required for the Athena Vortex-Lattice (AVL) code
  - add `VIEW = 4` to create a built-up-element-model (BEM) for a finite-element computation