Computational Aircraft Prototype Syntheses (CAPS) Training
Session 3

John F. Dannenhoffer, III
Syracuse University

Bob Haimes
Massachusetts Institute of Technology

caps
Overview

- Muddy cards
- Review of session 2 exercises
- Analysis of wing structure
  - required Bodys
  - required attributes (naming vs. meta-data)
  - dissection of wing3.csm
- Structural analysis
  - dissection of astrosModal.py
  - dissection of astrosStatic.py
New Design Parameters for wing3

- wing:spar1 0.20 location of fwd spar
- wing:spar2 0.70 location of rwd spar
- wing:nrib 3.00 number of ribs per wing
Required Bodys (for Structural Analyses)

- Outer Mold Lines (OMLs) for each component
  - FuseOml (a SolidBody)
  - WingOml (a SolidBody)
  - HtailOml (a SolidBody)
  - VtailOml (a SolidBody)

- Waffle for each component
  - FuseWaffle (a SheetBody) — not yet supported
  - WingWaffle (a SheetBody)
  - HtailWaffle (a SheetBody) — not yet supported
  - VtailWaffle (a SheetBody) — not yet supported
Required Attributes of WingWaffle

- **Body**
  - (none required)

- **Faces**
  - `tagComp` with value `$leftwing`, `$riteWing`, or `$wing` (if on symmetry plane)
  - `tagType` with value `$spar` or `$rib`
  - `tagIndex` with different value for each spar and rib
Called with `.csm` statement:
```
UDPRIM waffle depth <number> filename <name_of_file>
```

Valid statements in file are:
- **CPOINT** — create a construction point (not in final waffle)
- **CLINE** — create a construction line (not in final waffle)
- **POINT** — create a waffle point
- **LINE** — create one or more waffle segments
- **PATBEG/PATEND** — create a pattern (loop)

Keywords can be in lowercase or UPPERCASE.

Coordinates of existing point `<pname>` are given by
- `x@<pname>` and `y@<pname>`
Variants of CPOINT and POINT

- POINT <pname> AT <xloc> <yloc>
  - create point at <xloc>,<yloc>
- POINT <pname> ON <lname> FRAC <fracDist>
  - creates point on <lname> at given fractional distance
- POINT <pname> ON <lname> XLOC <x>
  - creates point on <lname> at given <x>
- POINT <pname> ON <lname> YLOC <y>
  - creates point on <lname> at given <y>
- POINT <pname> ON <lname> PERP <pname2>
  - creates point on <lname> that is closest to <pname2>
- POINT <pname> ON <lname> XSECT <lname2>
  - creates point at intersection of <lname> and <lname2>
- POINT <pname> OFF <lname> <dist> <pname2>
  - creates point <dist> to the left of <lname> at <pname2>
Variants of CLINE and LINE

- LINE . <pname1> <pname2> <attrName1=attrValue1>...<br>
  creates unnamed line between <pname1> and <pname2> with given attribute(s) (if any)

- LINE . <pname1> <pname2> <attrName1=attrValue1>...<br>
  creates line named <lname> between <pname1> and <pname2> with given attribute(s) (if any)
Waffle for wing3 — 1

SET xmin @xmin-0.1
SET xmax @xmax+0.1
SET ymin 0
SET ymax @ymax+0.1
SET zmin @zmin-0.1
SET zmax @zmax+0.1
STORE .

UDPARG waffle depth wing:nrib # ensures rebuild
UDPARG waffle depth wing:spar1
UDPARG waffle depth wing:spar2
UDPRIM waffle depth zmax-zmin filename <<

# construction lines for spars
CPOINT A AT 0+wing:spar1*croot 0
CPOINT B AT wing_xtip+wing:spar1*ctip wing_ytip
CPOINT C AT 0+wing:spar2*croot 0
CPOINT D AT wing_xtip+wing:spar2*ctip wing_ytip

CLINE AB A B
CLINE CD C D
# rite spars
POINT E ON AB YLOC ymin
POINT F ON AB YLOC ymax
LINE EF E F tagComp=riteWing tagType=spar tagIndex=1

POINT G ON CD YLOC ymin
POINT H ON CD YLOC ymax
LINE GH G H tagComp=riteWing tagType=spar tagIndex=2

# rite ribs
PATBEG irib wing:nrib
   CPOINT I AT xmin wing_ytip*irib/(wing:nrib+1)
   CPOINT J AT xmax y@I
   LINE . I J tagComp=riteWing tagType=rib ...
      tagIndex=!val2str(irib,0)
PATEND
# left spars

POINT E AT x@E -y@E  
POINT F AT x@F -y@F  
LINE EF E F tagComp=leftWing tagType=spar tagIndex=1

POINT G AT x@G -y@G  
POINT H AT x@H -y@H  
LINE GH G H tagComp=leftWing tagType=spar tagIndex=2

# left ribs

PATBEG irib wing:nrib  
    CPOINT I AT xmin -wing_ytip*irib/(wing:nrib+1)  
    CPOINT J AT xmax y@I  
    LINE . I J tagComp=leftWing tagType=rib ...  
        tagIndex=!val2str(irib,0)  

PATEND

>>
Views of transport Waffle
Writing the `pyCAPS` File

- Preamble
- (Optionally) selecting Faces to ignore
- Defining the material and properties for the various Faces
- Defining the constraints
- Executing the preAnalysis
- Executing the solver
- Executing the postAnalysis

Note: many of these operations require the use of the attribute editor
Statements in the attribute editor can be one of:

- **NODE** <selector> <attrName1=attrValue1> ...
- **EDGE** <selector> <attrName1=attrValue1> ...
- **FACE** <selector> <attrName1=attrValue1> ...
- **AND** <selector> <attrName1=attrValue1> ...
- **ANDNOT** <selector> <attrName1=attrValue1> ...
- **SET** <attrName1=attrValue1> ...

Keywords can either be specified in lowercase or UPPERCASE.

<selector> can be one of HAS, ADJ2NODE, ADJ2EDGE or ADJ2FACE
Typical block of code looks like:

```plaintext
NODE ADJ2FACE tagType=spar tagIndex=1
AND ADJ2FACE tagType=lower
AND ADJ2EDGE tagType=root
SET capsConstraint=pointConstraint1
```

In the case, `capsConstraint` is the attribute that tells the ASTROS AIM that you are specifying a constraint.

Attribute editor is used by writing a `editAttr_<n>.txt` file, where `<n>` must be sequential, starting at 1.
Dissection of template_astrosModal.py

File can be found at $ESP_ROOT/training/CAPS/data

- Preamble
- (Optionally) selecting Faces to ignore (capsIgnore)
- Defining the material and properties for the various Faces (capsGroup)
- Defining the constraints (capsConstraint)
- Executing the preAnalysis
- Executing the solver
- Executing the postAnalysis
Start with configuration file (wing3.csm)

Start with template_astrosModal.py script for ASTROS

Change directory into $ESP_ROOT/training/CAPS/data

Run: python template_astrosModal.py wing3.csm
Modal Analysis Exercise

- Start with configuration file (wing3.csm)
- Make a copy of the original template_astrosModal.py script (and call it exercise_3_1.py)
- Modify exercise_3_1.py to create a plot of the first three modal frequencies as a function of spar thickness
  - you will have to import numpy and matplotlib
  - you will have to add a loop to run the various analysis
  - you will have to generate the plots (next page)
- Run: python exercise_3_1.py
Modal Analysis Exercise

Frequency change using ASTROS on wing3
Performing a Static Analysis

- Preamble
- (Optionally) selecting Faces to ignore (capsIgnore)
- Defining the material and properties for the various Faces (capsGroup)
- Defining the constraints (capsConstraint)
- Defining the loads (capsLoad)
- Executing the preAnalysis
- Executing the solver
- Executing the postAnalysis
First Static Analysis

ASTROS on wing3

- Start with configuration file (wing3.csm)
- Start with template_astrosStatic.py script for ASTROS
- Change directory into $ESP_ROOT/training/CAPS/data
- Run: python template_astrosStatic.py wing3.csm
Modify `template_astrosStatic.py` to plot the displacements on the left wing — upper and lower (next page)

Run: `python exercise_3_2.py`
Static Analysis Exercise

Left wing displacements using ASTROS on wing3
Muddy Cards

- Opportunity to provide immediate “feedback”
- Any questions about presentation material, critique of sample problems, . . .
- Questions will be answered at next session