Comparison of Automated and Hand-generated Overset Grids for Aerospace Applications

John F. Dannenhoffer, III
Syracuse University, Syracuse NY

Jeffrey P. Slotnick
Boeing Research & Technology, Huntington Beach CA

Darby Vicker and Reynaldo J. Gomez, III
NASA Johnson Research Center, Houston TX

Scott Sherer
Air Force Research Laboratory, Wright-Paterson AFB OH

12^{th} Symposium on Overset Composite Grids and Solution Technology

Overview

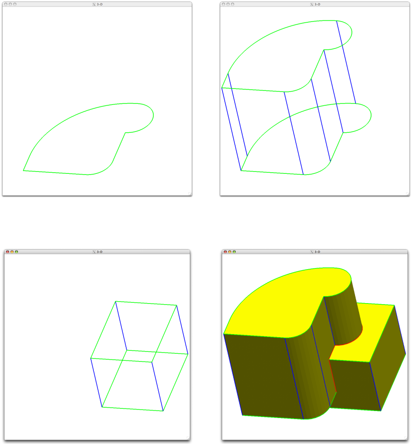
- Background & Objective — 2012
  - Review of technique & examples
- Background & Objective — 2014
- Trap wing with slat and flaps
  - Challenges
  - Overset grid system
  - Comparison with hand-tuned grids
- Launch escape vehicle
  - Challenges
  - Overset grid system
- High-lift test case
  - Challenges
- Summary

Background & Objective — 2012

- Background
  - overset grids can be used to produce high-accuracy, structured-grid viscous flow solutions
  - biggest limitation is the labor needed to:
    - decompose configuration into surface-sets on which to generate the component grids
    - identify regions in which collar grids need to be generated
  - fortunately, many configurations are currently modeled in a solid-modeling CAD system
- Objective
  - automate the overset grid generation process

Feature Tree Example

```
skbeg      0.0  0.0  0.0
   linseg  3.0  0.0  0.0
   cirarc  3.7  0.3  0.0  4.0  1.0  0.0
   linseg  4.0  3.0  0.0
   cirarc  5.0  4.0  0.0  4.0  5.0  0.0
   cirarc  1.2  3.8  0.0  0.0  1.0  0.0
skend

extrude    0.0  0.0  5.0

box        3.0  2.0 -1.0  3.0  .0  3.0

union

end
```

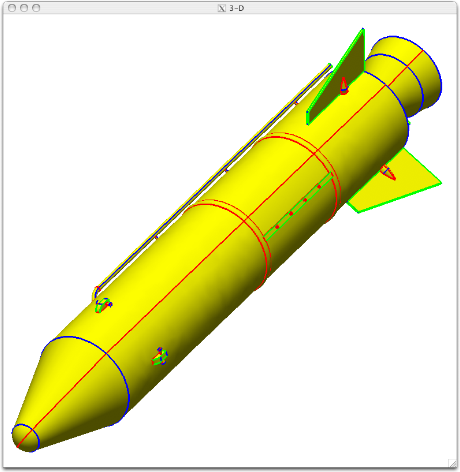


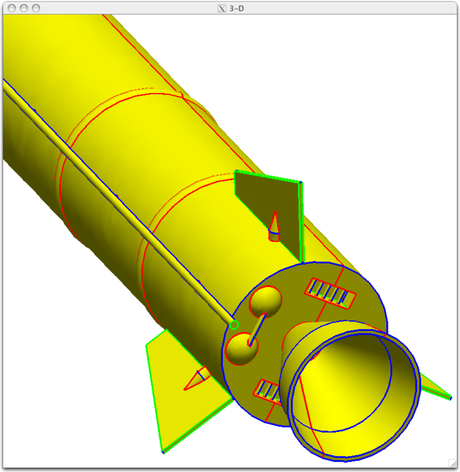
Overall Strategy

- Exploit duality between feature tree and overset grid system
  - for each primitive solid, generate basic grid(s)
  - for each Boolean spine, generate collar grid
  - generate global grid
  - cut holes & trim grids
  - set up donor information
- Implemented in the `OvrCad3` system
  - built upon `OpenCSM`, `EGADS`, and `OpenCASCADE`

Sample Configuration — JMR3

51 features, 58 primitive solids, 57 Boolean operators

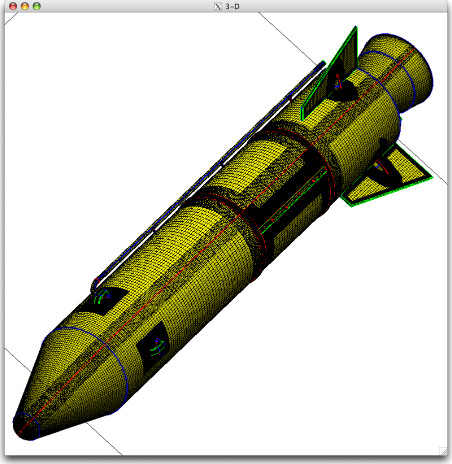


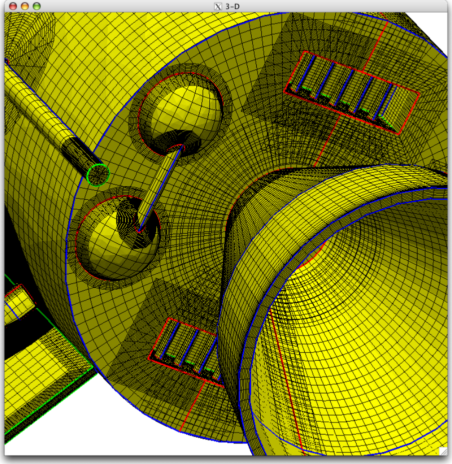


194 faces, 462 edges, 296 nodes

Surface Grids — JMR3

76 basic grids, 152 collar grids, 1 global grid

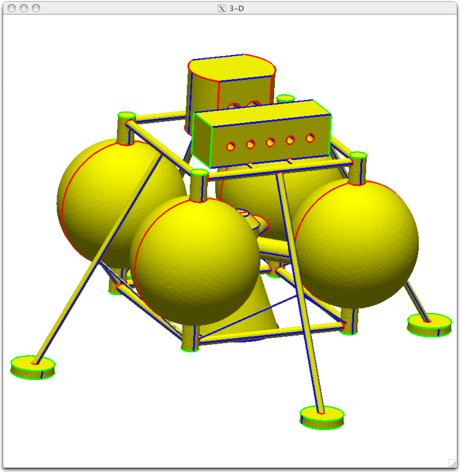


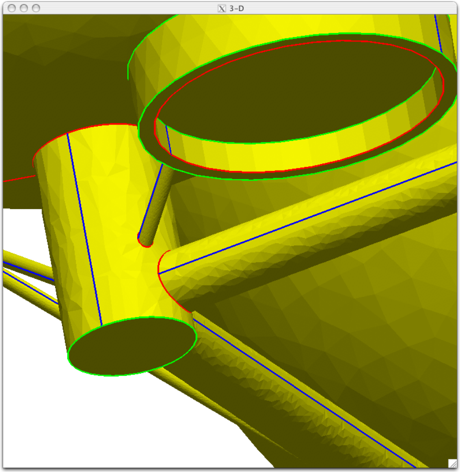


7 minutes on MacBook Pro 2.6 GHz Intel Core 2 Duo

Sample Configuration — Lander

53 features, 56 primitive solids, 55 Boolean operators

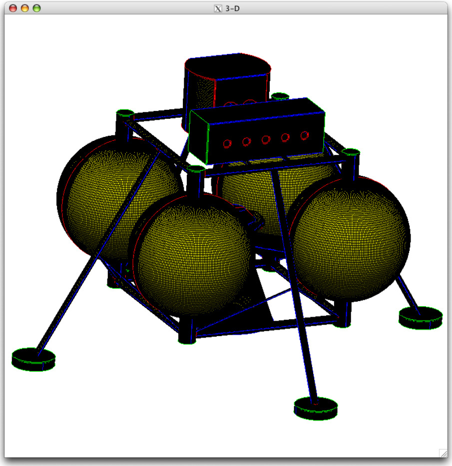


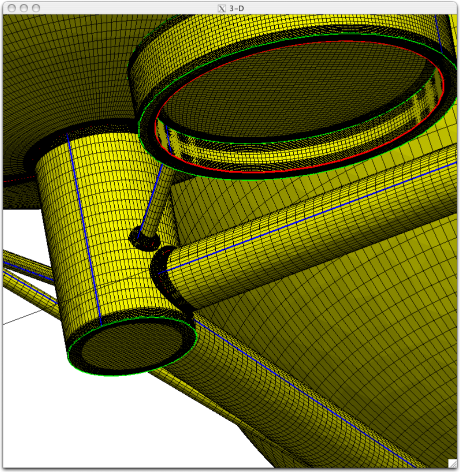


158 faces, 426 edges, 278 nodes

Surface Grids — Lander

74 basic grids, 114 collar grids, 1 global grid





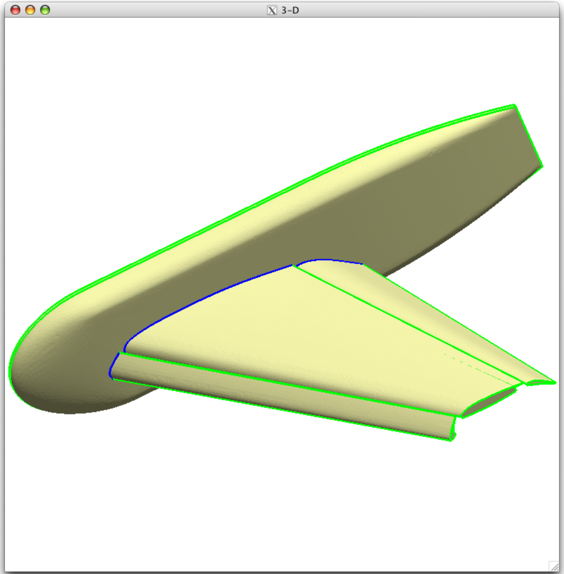
100 minutes on MacBook Pro 2.6 GHz Intel Core 2 Duo

Background & Objective —2014

- Background
  - After conference, several organizations wanted to evaluate for real-world applications and to know how good automatic grid systems were
    - Boeing — High-lift configurations
    - NASA/JSC — Launch abort vehicles
    - AFRL — Full transport configurations
- Objective
  - Compare "automated" with "hand-generated" grid systems
    - Time to generate
    - Computational efficiency
    - Computed accuracy

Trap Wing with Slat and Flap

- Defined in terms of 4 solids
  - slat
  - main
  - flap
  - half-fuselage



Bare-bones CSG (.csm) File

Trap Wing, Slat, Flap

```
# slat
import    $/trap_wing_slats_flaps.stp 1
   name        slat
   attribute   WingType       2         blunt TE with cove

# fuselage
import $/trap_wing_slats_flaps.stp 2
   name        fuselage
   attribute   FuseType       1         half fuselage in X

# move fuselage slightly outboard to insure successful unions
translate 0.0 -0.10 0.0
union

# flap
import $/trap_wing_slats_flaps.stp 3
   name        flap
   attribute   WingType       1         blunt TE
union

# main wing
import $/trap_wing_slats_flaps.stp 4
   name        main
   attribute   WingType       2         blunt TE with cove
union

end
```

Sample Attributes in `.csm` File

Slat from Trap Wing Configuration

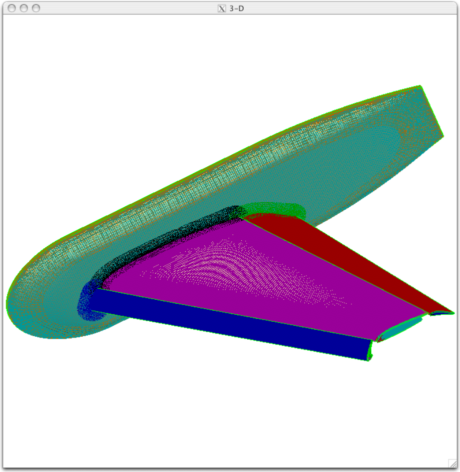
```
attribute   wing_kmax     65        number of points away from surface
attribute   wing_dnbeg    0.010     nominal off-body spacing
attribute   wing_dnrat    1.150     normal stretching ratio

attribute   wrap_nte      4
attribute   wrap_jtrm1    2
attribute   wrap_jtrm2    8
attribute   wrap_dsimin   0.0025    times Croot or Ctip
attribute   wrap_dsimax   0.010     times Croot or Ctip
attribute   wrap_dsjmin   0.0005    times Bsemi
attribute   wrap_dsjmax   0.010     times Bsemi

attribute   clamp_imax    151
attribute   clamp_itrm1   2
attribute   clamp_itrm2   2
attribute   clamp_jmax    41
attribute   clamp_j1      12
attribute   clamp_j2      28
attribute   clamp_dsjfac  0.250     times nominal dsi
attribute   clamp_dsjrat  1.150

attribute   corn_imax     49
attribute   corn_i1       24
attribute   corn_dsibeg   3.000     times TE spacing
attribute   corn_dsirat   1.150
attribute   corn_jmax     57
attribute   corn_j1       24
attribute   corn_j2       32
attribute   corn_dsibeg   1.000     times nominal dsi
```

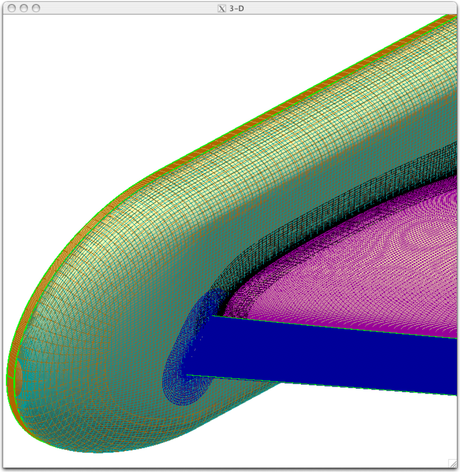
Trap Wing — Challenges

- ✓ 4 solids “almost” intersected
- ✓ Primitive grids associated with “wing” solids
  - wrap-around O-type grid
  - clamp grid over most of tip, with automatic placement of upper-surface points for wings with coves
  - cuff (collar) grid near tip leading edge
  - corner grid near tip (blunt) trailing edge
- ✓ Primitive grids associated with “fuselage” solids
  - wrap-around grid for most of fuselage
  - special collar-like grid adjacent to symmetry plane
- ✓ `OvrCad3` attributes consistent with Boeing “gridding guidelines”
- ✓ Adjustment of `surgrd` and `hypgen` inputs to work for all grids
- ✓ Generation of surface grids only does not expose problems in generating interpolation stencils

Trap Wing, Surface Grids — 1

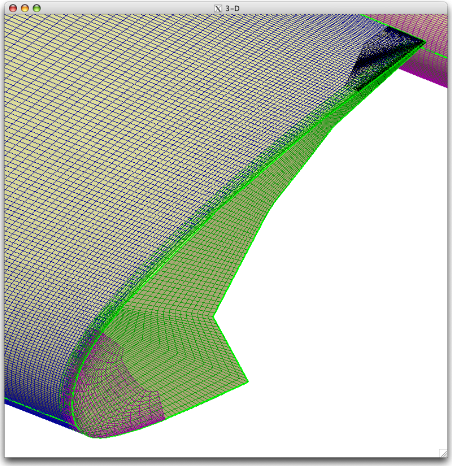
17 grids, 360,000 surface points

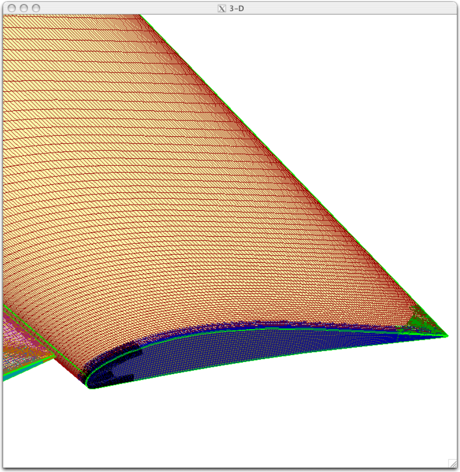




Trap Wing, Surface Grids — 2

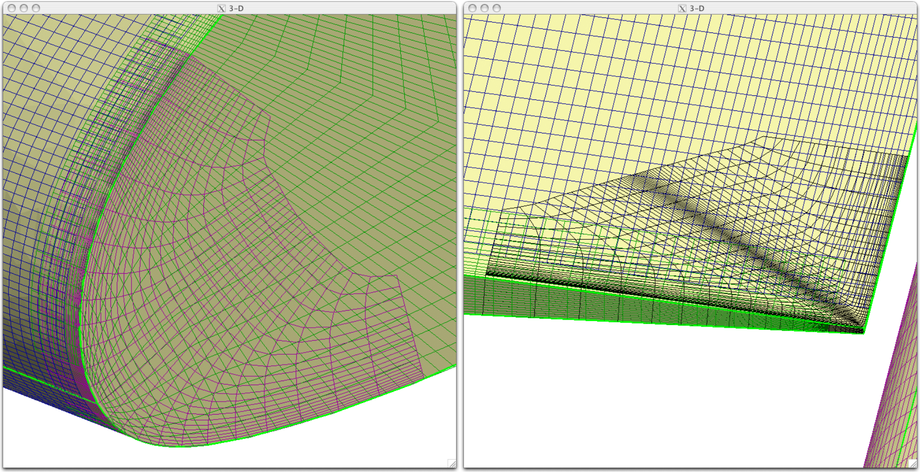
17 grids, 360,000 surface points





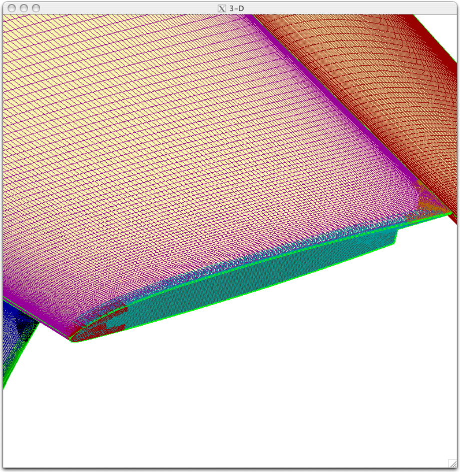
Trap Wing, Surface Grids — 3

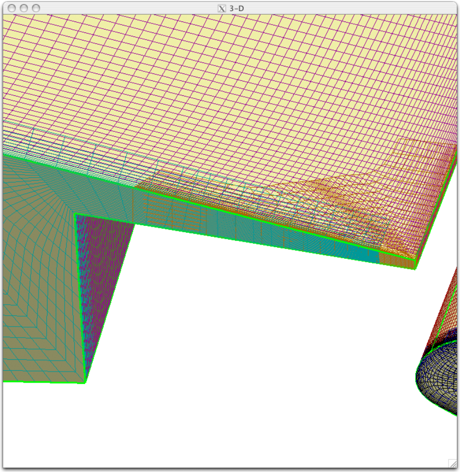
17 grids, 360,000 surface points



Trap Wing, Surface Grids — 4

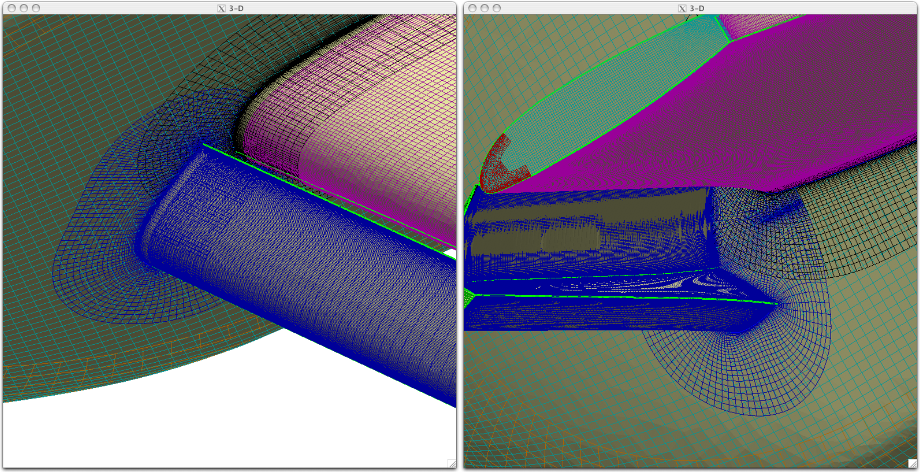
17 grids, 360,000 surface points





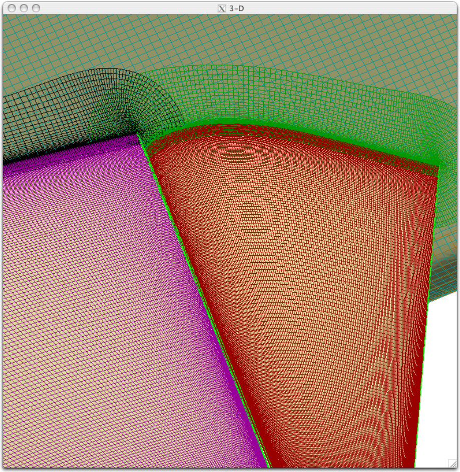
Trap Wing, Surface Grids — 5

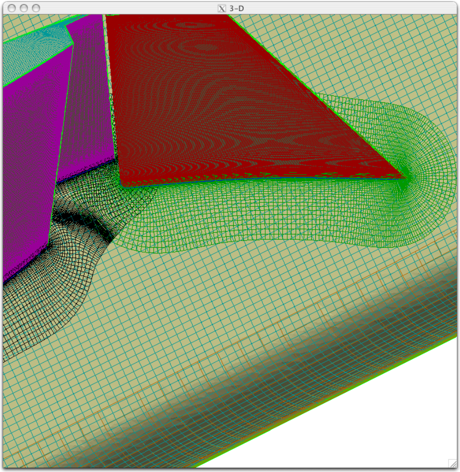
17 grids, 360,000 surface points



Trap Wing, Surface Grids — 6

17 grids, 360,000 surface points





Labor Required

Trap Wing, Slat, and Flap

Phase	Hours
Generation of CSG model	4
Generation of `OvrCad3` attributes	
$\longrightarrow$ initial	2
$\longrightarrow$ each of 10 iterations	1
Total	16

CPU Time Required

Trap Wing, Slat, Flap

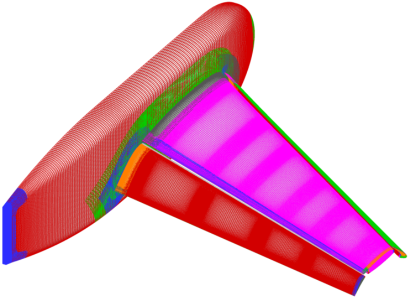
Phase	CPU sec	%
Build BRep	72	8
Tessellate	23	2
Generate basic grids	338	36
Build collar grids	280	30
Overhead	222	24
Total	935	100

CPU times on MacBook Pro 2.6 GHz Intel Core 2 Duo computer

Comparison with Hand-generated Grids

Trap Wing, Slat, and Flap

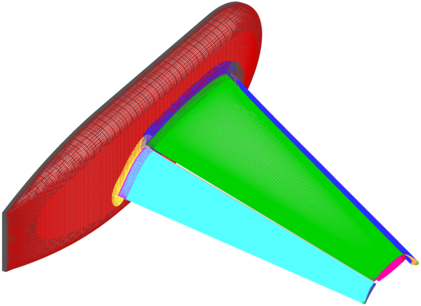
Hand-generated (Boeing)



80 hours

34 zones, 47.0M grid points

OvrCad3

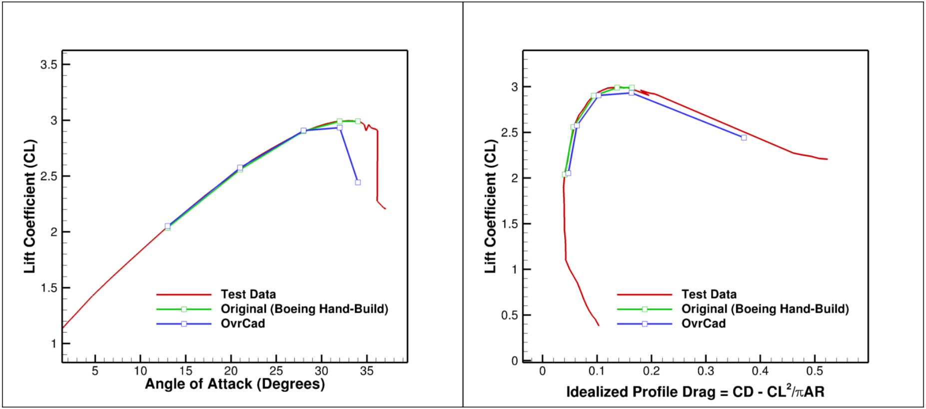


16 hours

19 zones, 27.2M grid points

Comparison with Hand-generated Grids

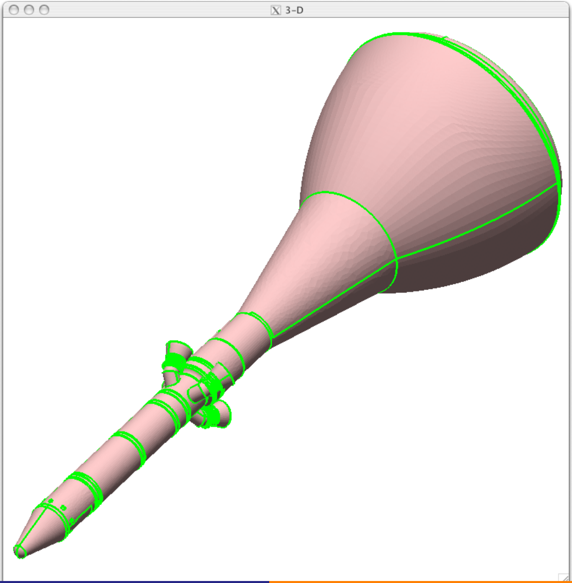
Trap Wing, Slat, and Flap



all calculations done without adaptation

Launch Escape Vehicle

- Defined as Pro/ENGINEER model
- 1 solid made up of 234 connected faces
- Feature tree not CSG



Bare-bones CSG (.csm) File — 1

Launch Escape Vehicle

```
# original Body from .stp file
import      $/75aa_cfd_wo_jettison_motors.egads
   attribute  SkipGrid        1
   attribute  ShowBody        0

# main body
skbeg          0        0          0
   cirarc    35        97          0       108       146          0
...
   cirarc 15810      1310          0     15957         0          0
   linseg      0        0          0
skend
revolve        0        0          0   1   0   0       360

...
```

Bare-bones CSG (.csm) File — 2

Launch Escape Vehicle

```
...

# ejection nozzles (each in two pieces)
patbeg i 4
   mark

   udprim    ellipse    rx  156    rz   156
   rotatez   -26.8      0          0
   translate 5714       311        0

   udprim    ellipse    rx  165    rz   156
   rotatez   -44.6      0          0
   translate 5835       550        0

   udprim    ellipse    rx  156    rz   156
   rotatez   -65.3      0          0
   translate 6007       629        0

   rule

   skbeg                0         0         0
      linseg          190         0         0
...
      linseg            0         0         0
   skend
   revolve           5750       200         0    0  1  0       360

...
```

Bare-bones CSG (.csm) File — 3

Launch Escape Vehicle

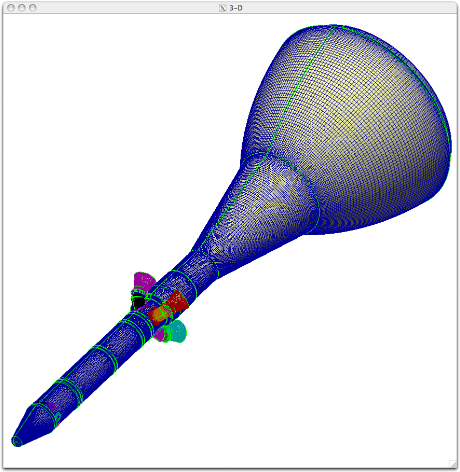
```
...

   union
   rotatex          68+i*90      0          0
   union
patend

# steering motors
patbeg i 8
   cylinder         1345       380          0      1345       415          0         45
   rotatex          i*45         0          0
   subtract
patend

end
```

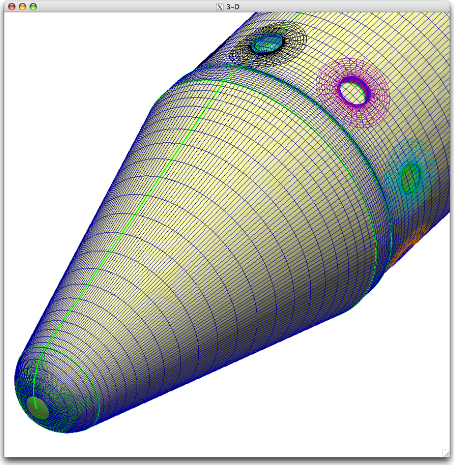
Launch Escape Vehicle – Challenges

- ✓ Create “idealized” constructive solid model
  - created “idealization” with 17 primitives
- ✓ Primitive grids associated with “body of revolution” solids
  - wrap-around grid
  - cap grids (near axis of revolution)
- ✓ Special treatment in cavity near heat-shield
- ✓ “Morph” grid from idealization to actual model

Launch Escape Vehicle, Surface Grids — 1

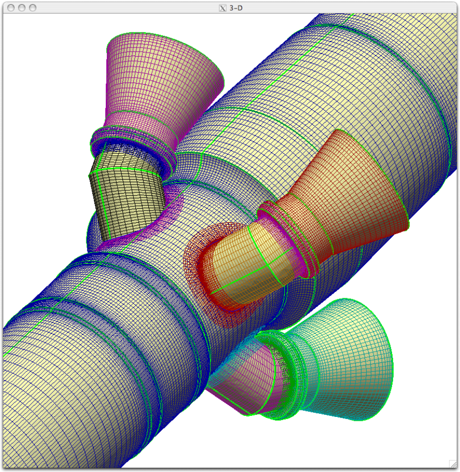
23 basic grids, 16 collar grids, 6.21M grid points

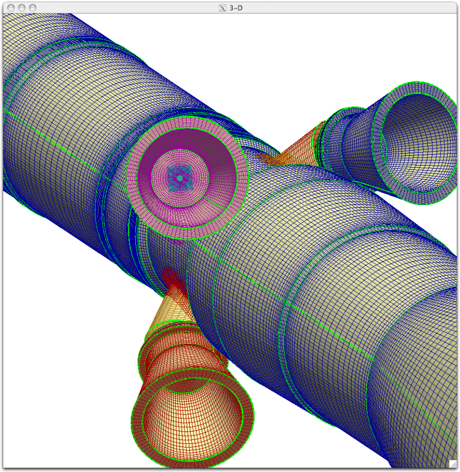




Launch Escape Vehicle, Surface Grids — 2

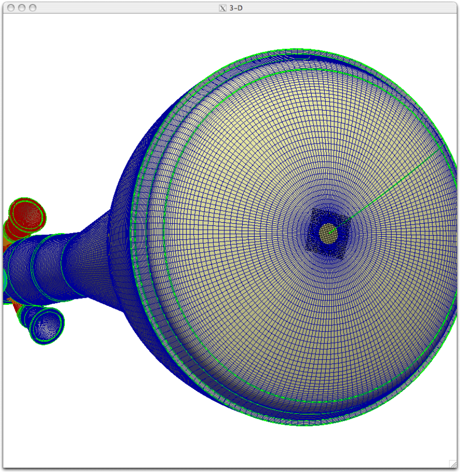
23 basic grids, 16 collar grids, 6.21M grid points





Launch Escape Vehicle, Surface Grids — 3

23 basic grids, 16 collar grids, 6.21M grid points



Labor Required

Launch Escape Vehicle

Phase	Hours
Generation of CSG idealization	4
Association of idealization to geom.	8
Generation of `OvrCad3` attributes	
$\longrightarrow$ initial	2
$\longrightarrow$ each of 2 iterations	1
Total	16

CPU Time Required

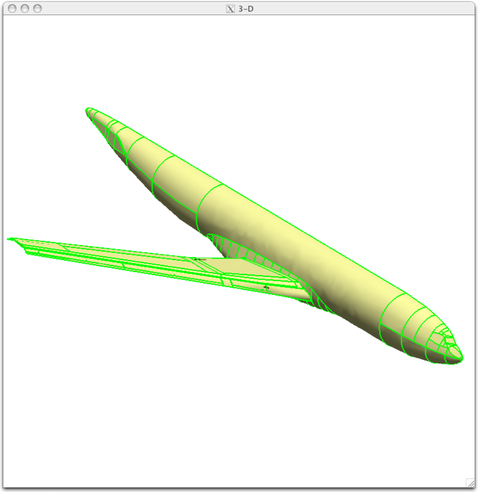
Launch Escape Vehicle

Phase	CPU sec	%
Build BRep	117	31
Tessellate	132	35
Generate basic grids	4	1
Build collar grids	96	26
Overhead	25	7
Total	374	100

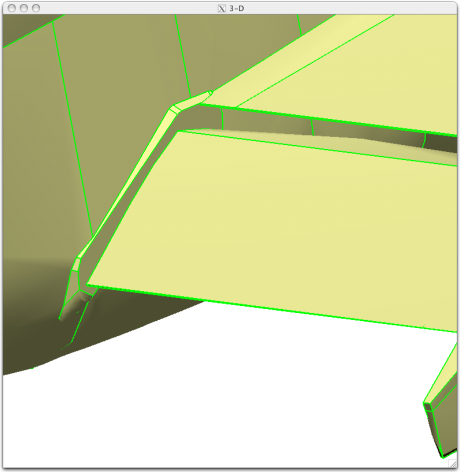
CPU times on MacBook Pro 2.6 GHz Intel Core 2 Duo computer

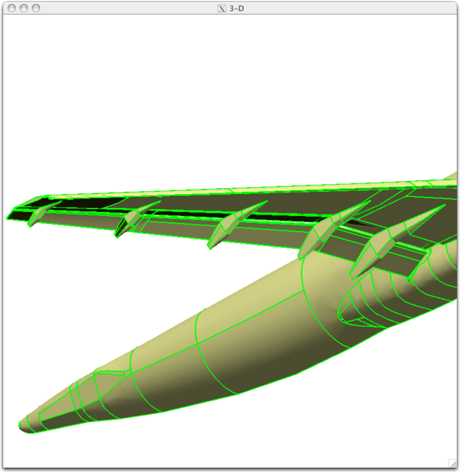
High-lift PW2 Test Case

- Defined in terms of a bag of 477 unconnected faces
- More features than Trap Wing
  - flap track fairing
  - features on fuselage



High-Lift PW2 Test Case





High-lift PW2 — Challenges

- × Idealized model with appropriate features
- × Primitive grids associated with fairing, . . .
- × "Morph" grid from idealization to actual model
  - morphing does not currently move edges

Summary

- Application of `OvrCad3` to "real-world" configurations was not easy
  - most legacy configurations are not defined as constructive solid models
  - difficult to "match" traditional best practices with an automated tool that has limited "settings"
- Once all the necessary pieces are in hand, `OvrCad3` can generate overset grid systems in a matter of minutes
- Preliminary results indicate that:
  - labor to create grids greatly reduced
  - automatic grid a bit larger than hand-generated grid because attributes were set conservatively
  - results computed with `OvrCad3` have drag error at high angles of attack
    - global and near-body adaptation might fix wake spacing

Future Directions

- `OvrCad3` development and application to real-world problems will continue as part of AFRL CAPS project
  - Computational Aircraft Prototype Syntheses
  - Geometry discretization objectives:
    - Generate discretization rapidly that can be used for aerodynamic prediction methods that do not require a volume mesh
    - Discretization of the structures (including support for FSI)
    - Support solvers that require an unstructured, body-fitted mesh
    - Support solvers that require structured overset meshes
    - Support of high-order meshes