

Engineering Sketch Pad (ESP)



Training Session 13 Associated Tools

John F. Dannenhoffer, III

jfdannen@syr.edu

Syracuse University

updated for v1.25

- Other `.csm` statements
- CAPS (described earlier)
- ErepEd
- Plugs
- Pyscript (described earlier)
- VspSetup
- Homework exercise

- `APPLYCSYS $csysName ibody=0`
 - transforms Group on top of stack so that their origins/orientations coincide with given csys
- `ASSERT arg1 arg2 toler=0 verify=0`
 - return error if arg1 and arg2 differ
- `CHAMFER radius edgeList=0`
 - apply a chamfer to a Body
- `EVALUATE $type arg1 ...`
 - evaluate coordinates of NODE, EDGE, or FACE
- `FILLET radius edgeList=0 listStyle=0`
 - apply a fillet to a Body

- GETATTR \$pmtrName attrID global=0
 - store an Attribute value(s) in a LOCALVAR
- HOLLOW thick=0 entList=0 listStyle=0
 - hollow out a SolidBody or SheetBody
- MESSAGE \$text \$schar=_ \$fileName=. \$openType=a
 - generate a message to be displayed
- PROJECT x y z dx dy dz useEdges=0
 - find the first projection from given point in given direction
- SOLBEG / SOLCON / SOLEND
 - solve simultaneous non-linear equations

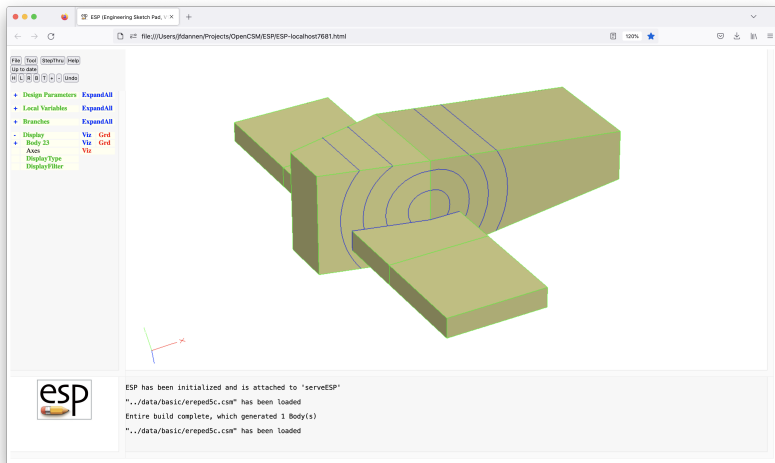
- An Erep is an effective topology that may be used in a down-stream analysis (such as CAPS)
- Ereps share geometry with a supporting Brep (which is comprised of Nodes, Edges, and Faces)
- ENodes in an Erep are a subset of the Brep's Nodes
- EEdges in an Erep are combinations of the Brep's Edges
 - the EEdges are the concatenation of one or more contiguous Edges
 - note: not all Edges are associated with an EEdge (that is, some Edges may be interior to an EFace)
- EFaces are combinations of one or more of the Brep's Faces
 - note: every Face is associated with an EFace

- Ereps are specified via attributes on the Brep
- Nodes that have a `.Keep` Attribute are forced to be ENodes
- Edges that have a `.Keep` Attribute are forced to be part of an EEdge
- Attributes on the Faces define which ones are to be combined into an EFace
 - The Attribute name is user-selected (for example `_erep`)
 - All Brep Faces that have a `_erep` Attribute with the same integer value are candidates to be combined into an EFace
 - the Faces must be contiguous
 - the dihedral angle between adjacent Faces must not exceed a user-specified tolerance (typically 5 degrees)
 - The Erep Editor uses “colors” to specify the Attribute values



Erep Editor (3)

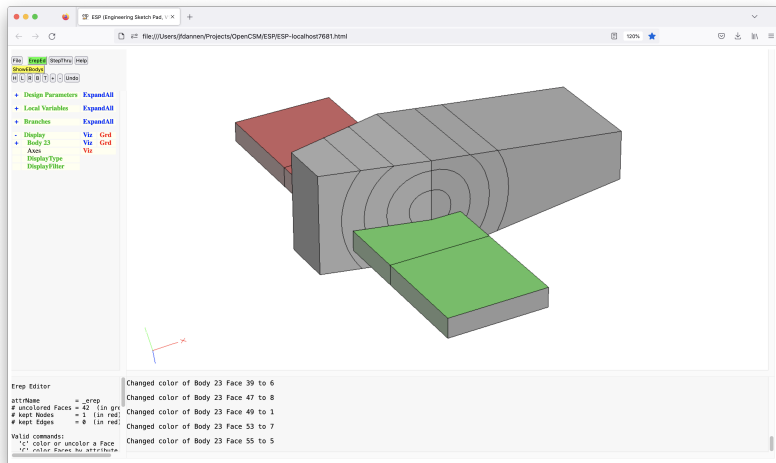
Original Brep configuration





Erep Editor (4)

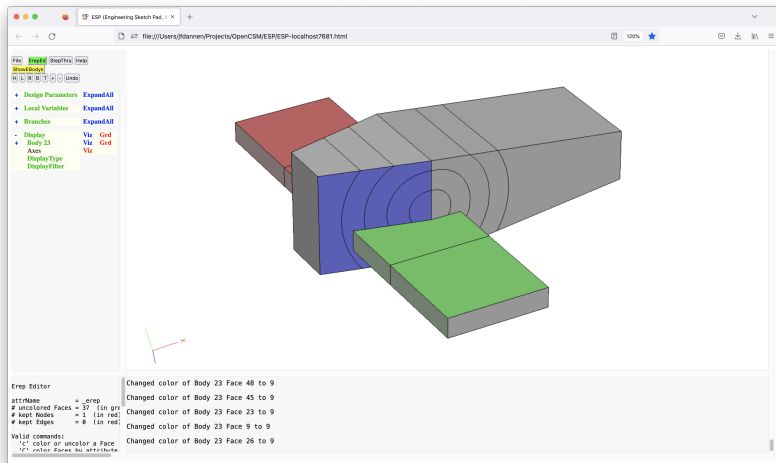
Wing Faces colored into 8 groups (using the “C” option)





Erep Editor (5)

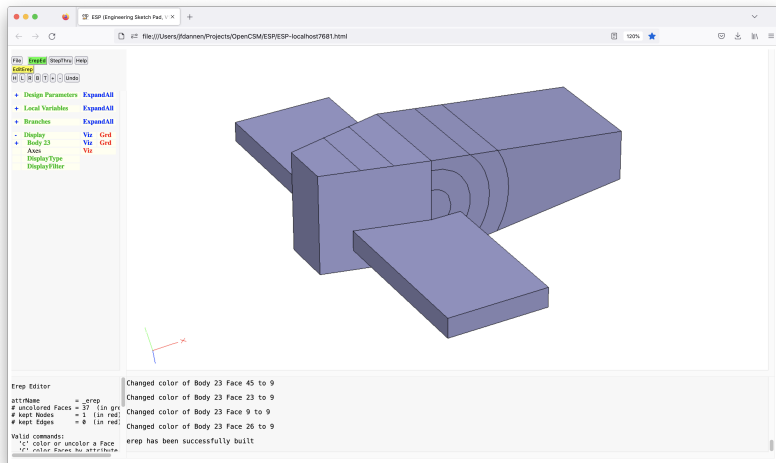
Fuselage side colored into one group (using the “c” option five times)





Erep Editor (6)

Associated Erep



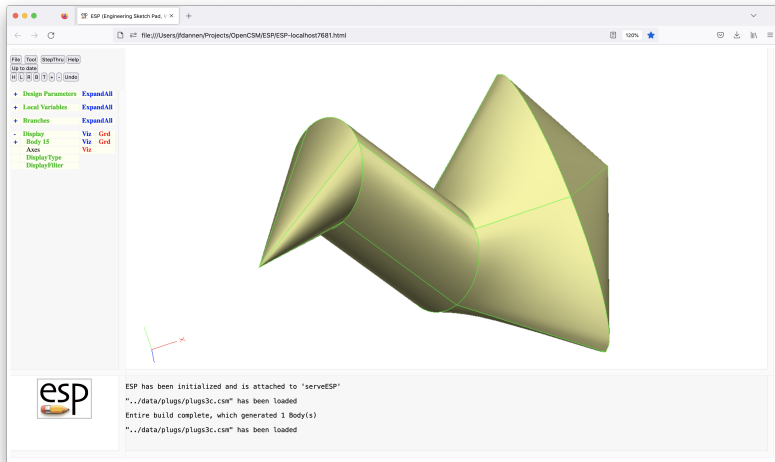
- **Plugs** is a tool for finding the Design Parameter values such that a Body matches a cloud of points
- Inputs:
 - a single parameterized Body
 - a cloud of (unclassified) points
- Outputs:
 - Design Parameter values
- Executed via **Tool** → **Plugs**
- Algorithm is described in Jia, P, and Dannenhoffer, J.F., “Generation of Parametric Aircraft Models from a Cloud of Points”, AIAA-2016-1926, presented at AIAA SciTech 2016, January 2016.

- Phase 1
 - Take up to 50 Levenberg-Marquardt optimization steps to vary the Design Parameters to match the bounding box of the point cloud
 - Unclassify all points in the cloud
- Phase 2
 - In a sequence of passes
 - classify each point in the cloud by determining the most likely Face to associated it with; if there is no obvious Face, leave the point unclassified
 - if all points are classified and the point classifications are the same as in the previous pass, exit
 - perform up to 50 steps of the Levenberg-Marquardt optimizer



Plugs (3)

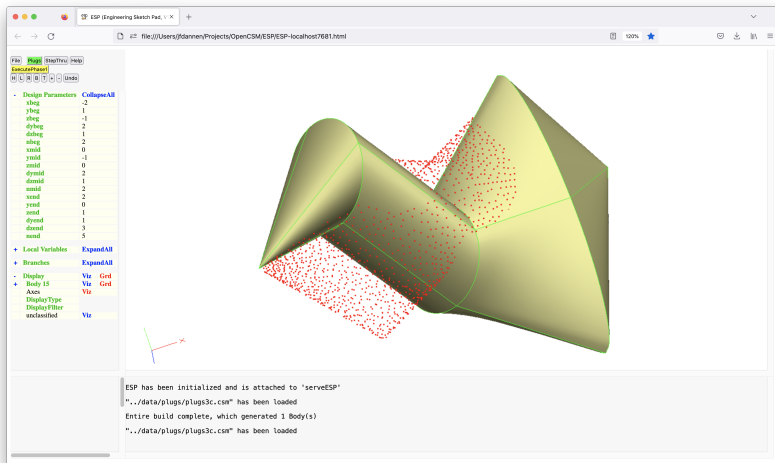
Initial Body with user-guessed Design Parameters

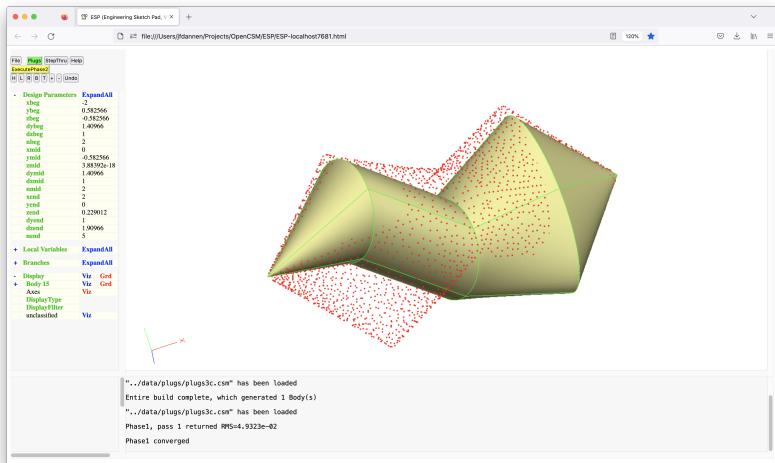




Plugs (4)

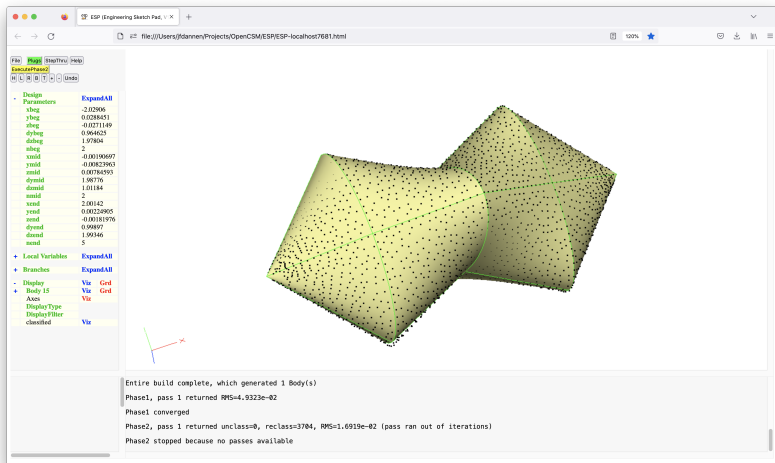
Initial Body with cloud of points (red points are unclassified)

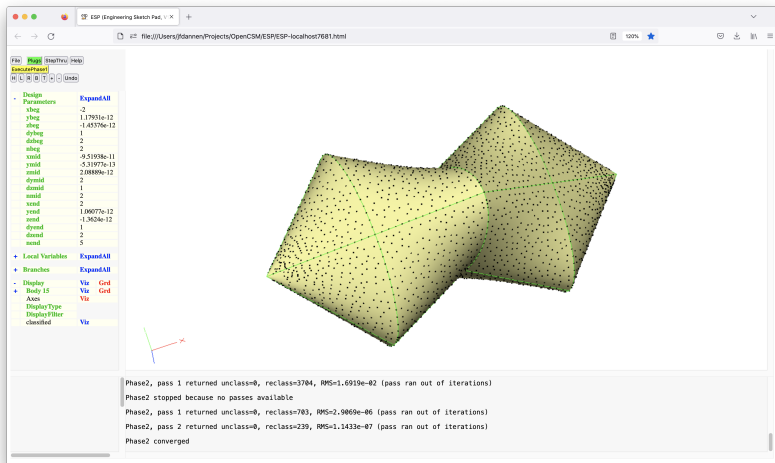




Plugs (6)

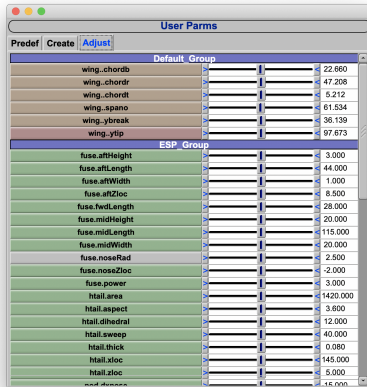
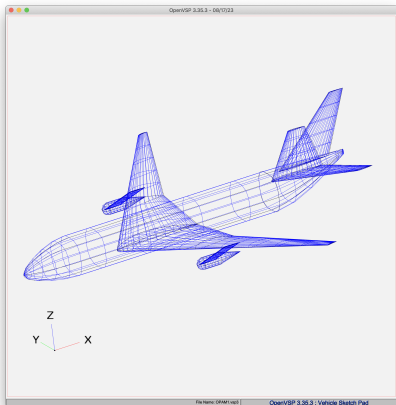
After Pass 1 of Phase 2 (black points are classified)





- The `vsp3` UDP can be used to import an `OpenVSP` model into ESP
- Each component in the `OpenVSP` model comes into ESP as a separate `SolidBody`
- The `SolidBodys` are static (i.e., they have fixed geometry)
- In order to drive an `OpenVSP` model parametrically, one needs to use the `VspSetup` tool
 - creates a `.udc` that defines the **UserParms** in `OpenVSP`
 - calls the `vsp3` UDP
- In order to use the `vsp3` UDC, either `vspscript` must be in the user's path or the user must set the `VSP3_ROOT` environment variable to the directory containing `vspscript`

- The `.vsp3` file is generated by `OpenVSP` and must exist before the process starts
- `OpenVSP` model should have “User Parms”
 - in the “ESP_Group”
 - names include periods for grouping



- Every time the user modifies the `.vsp3` file:
 - in `serveESP`, use **Tool**→**VspSetup**, to write a `.udc` file that contains:
 - DESPMTR, LBOUND, and UBOUND statements from the tagged variables in the `.vsp3` file
 - a UDPRIM `vsp3` statement to generate an updated configuration (via a call to `vspscript`)

```
# generated by VspSetup from ../data/vsp3/OPAM1.vsp3
```

```
INTERFACE . ALL
```

```
DESPMTR    fuse:aftHeight    3
LBOUND     fuse:aftHeight    -100000
UBOUND     fuse:aftHeight    100000
```

```
<lines deleted>
```

```
DESPMTR    wing:zroot        -5
LBOUND     wing:zroot        -100000
UBOUND     wing:zroot        100000
```

```
UDPRIM     vsp3      filename $$/OPAM1.vsp3
```

```
END
```

- Write a `.csm` file that contains
 - a UDPRIM `/udcName` to include the `.udc` file

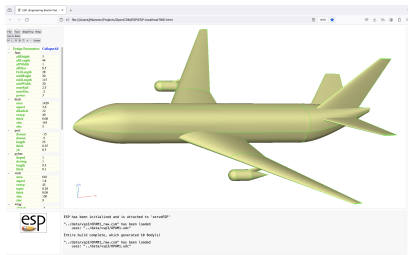
```
# import the geometry
UDPRIM $/OPAM1
```
 - any other `csm` statements, such as Boolean operations or specifying attributes

```
# make a single Body
UNION 1

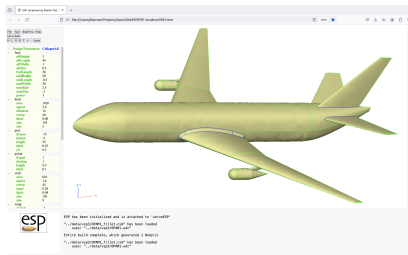
# add FILLETs at the wing/fuselage junctions
DESPMTR filrad 1.5

SELECT EDGE    1 0 3 0 0
FILLET filrad @sellist 1

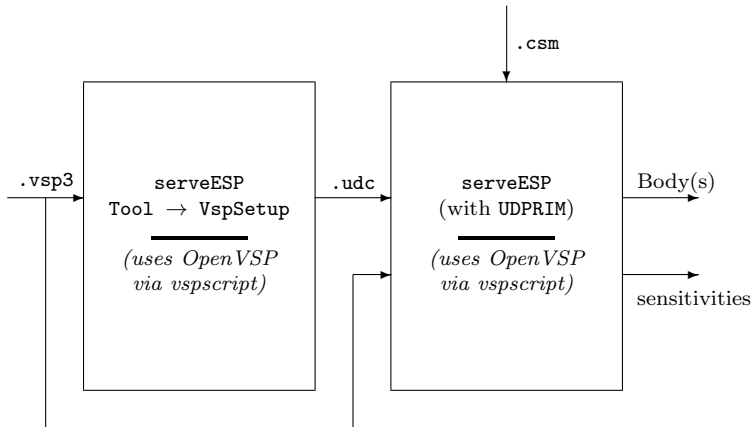
SELECT EDGE    2 0 3 0 0
FILLET filrad @sellist 1
```
- Changes to the DESPMTRs in ESP are reflected in changes to the geometry
- When the configuration is built in `serveESP`, geometric- and tessellation-sensitivities are available



Bodys flying
in flying formation



Bodys UNIONEd and
FILLETs added

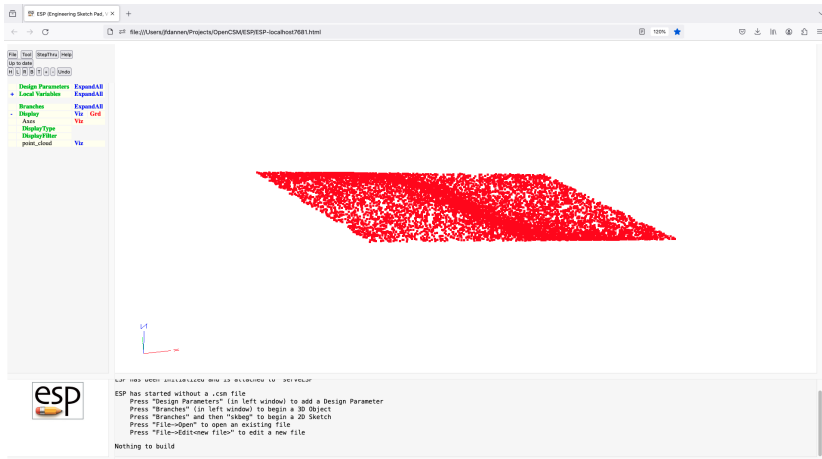


- Run `serveESP` with `plate.cloud` as the plotfile to view the points that should be fitted
- Write `plate_setup.csm` that creates a BOX whose top Face approximately corresponds to the points in `plate.cloud`
 - put the `_pluginsIgnore` attribute on all Faces except the Face that corresponds to the points in `plate.cloud`
 - convert the box to Bsplines by using the `nuscale` UDF
 - DUMP the file `plate.egads`
- Write `plate_plugs.csm` that will be used by Plugs
 - set up a `DESPMTR` that is as big as the number of internal control points and whose initial values are 0
 - use the `warp` UDF to warp the Face
- Run **Tools→Plugs**



Plugs (1)

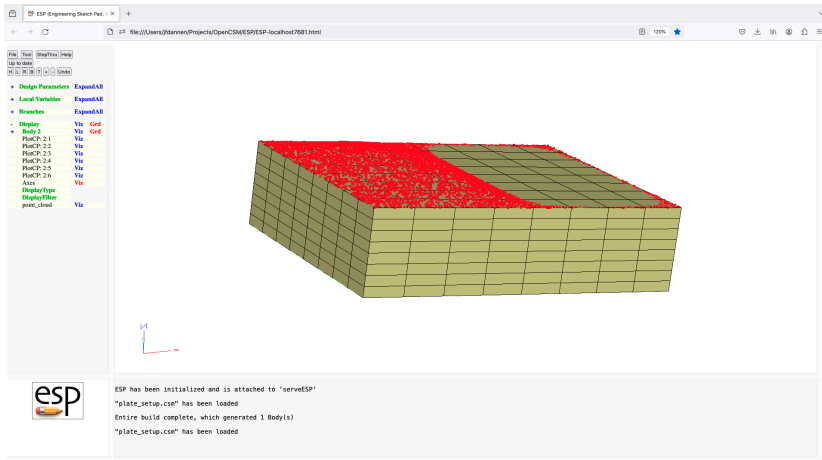
Original point cloud





Plugs (2)

Original configuration





Plugs (3)

Final configuration

