The Role of Geometry in MDO

John F. Dannenhoffer, III*

Aerospace Computational Methods Laboratory

Syracuse University, Syracuse, New York, 13244

As multi-disciplinary optimization (MDO) environments are applied to the design of complex vehicles, one must consider both the traditional scalar data, such a design parameters and integrated performance measures, and the actual surface shapes of the vehicle. Unfortunately, the analyses used in an MDO study all require different representations of the configuration. Creating these representations by reverse-engineering a manufacturing-based computer-aided design (CAD) model is extremely difficult and very prone to errors. Also, in order for the optimization to be efficient, especially with hundreds or thousands of design parameters, the optimization tool of choice is typically gradient based. Many recent analysis tools include an adjoint capability to make the computation of the gradients efficient; this then requires that the geometry system be somehow included in the gradient computations.

Key to creating models for MDO are three enabling technologies. First, one needs to create a series of geometric "views" of a configuration (one for each disciplinary analysis), that are driven from a single set of design parameters and which are linked to each other via rich attribution. Second, there must be a method for attaching data, such as pressures and displacements, to the various linked models and then transferring the data between the models via simple interpolation or via a transfer that conserves the integral and moments of the data. And third, the resulting system should provide analytic sensitivities of the configuration and its tessellation with respect to the user-defined design variables; this capability allows one to close the adjoint loop needed with gradient-based optimization.

The Engineering Sketch Pad (`ESP`) is a system that meets all these requirements. It is a CAD-like system for the generation of geometric models for the analysis and design of complex configurations, such as aircraft. It is a feature-based, parametric solid modeler that has been coupled directly to large number of fluid-thermal-structural-control analysis programs via the Computational Aerospace Prototype Synthesis (`CAPS`) system.

This paper describes a general architecture for incorporating geometry into a MDO framework and then demonstrates this architecture by applying `ESP` to a transport configuration.

I. Background

The multi-disciplinary optimal design (MDO) of an aerospace vehicle, such as an aircraft, requires that several domain analyses be executed, and that their results be communicated with some controlling MDO framework or shared amongst each other. In almost all cases, the domain analyses use geometric representations of the vehicle.

Amongst the most common MDO frameworks that are used in the aerospace community are ModeFRONTIER,[1] ModelCenter,[2] and OpenMDAO.[3] None of these have a built-in geometry modeling facility, but instead rely on linkages to commercial computer-aided design (CAD) packages. To use the CAD package, it is assumed that the user has developed a (single) parametric CAD model of the vehicle being designed.

*Associate Professor, Mechanical and Aerospace Engineering, AIAA Associate Fellow.

Then the MDO interacts with CAD by setting the design parameters that describe a specific instance of the vehicle, causing the CAD system to "rebuild" and then using the output of the CAD system, generally in the form of an IGES or STEP file of a surface tessellation. In some cases the CAD representation is reverse-engineered to develop the geometric representation that a discipline's analysis requires.

When using gradient-based optimization, one needs to be able to compute the sensitivity of the optimization objective function and constraints with respect to the design parameters. Since none of the commercially-available CAD systems provide such sensitivities, finite differences are used to compute the required sensitivities; this process is fragile since (1) often parameter changes can result in subtle topology changes in the geometric model, and (2) it is often difficult to determine the relationships of the topological elements in the base and perturbed representations.

II. System Requirements and Design

This section begins with a description of the various requirements that a system that puts geometry at the heart of the MDO process are described. This then leads to a system architecture that satisfies these requirements.

A. Requirements

In order for geometry to be central to the MDO process, it must provide many "views" of the geometry, each one of which is tailored to a specific discipline and fidelity. The need for these views is often because different analyses focus on different aspects of the design (that is, aerodynamics focuses on the overall shape of the wing whereas structural analyses focus on the various skin panels). But in addition, the idiosyncrasies of the various analyses require that the geometry be defined in different ways (that is, surface shape or cross-sections).

To be useful in MDO, the various views need to be driven from a common set of design parameters. Some of these design parameters define the overall shape (such as the wing planform) and others describe more detailed information (such as spar locations or number of ribs). Keys to defining good design parameters are that they must (1) be consistent with the way engineers describe the configuration and (2) must be complete, but not redundant. (For example, if one uses wing area and aspect ratio as design parameters, the wing span must not also be used since it can be directly computed from the former.)

The system must support a rich attribution scheme that allows the parts of the various views to communicate with each other. These attributes, which are user-defined name-value pairs (where the values can be vectors or strings) can be used to both describe the various pieces of the geometry as well as a method for prescribing boundary conditions and/or mesh characteristics.

In order to be useful with a gradient-based optimization scheme, the system must provide the sensitivity of the shape with respect to any (or all) of the design parameters. While it is true that non-gradient-based optimization methods are useful for early design space exploration, most users have found that gradient-based methods are much more efficient, especially once the overall configuration has been established.

B. System Design

A system architecture that satisfies the above requirements is shown in Fig. 1. While this architecture is not unique, it represents one such architecture that has been found to be useful for a variety of MDO studies.

C. Component models

At the bottom of Fig. 1 are the **component models**, which are the various shapes that make up the configuration. These shapes, which are driven by a common set of design parameters, represent not only the outer shape of the components, but also shapes for the various internal components. Fig. 2 shows some of the various component models for a wing of a transport configuration, such as the outer mold line (OML).

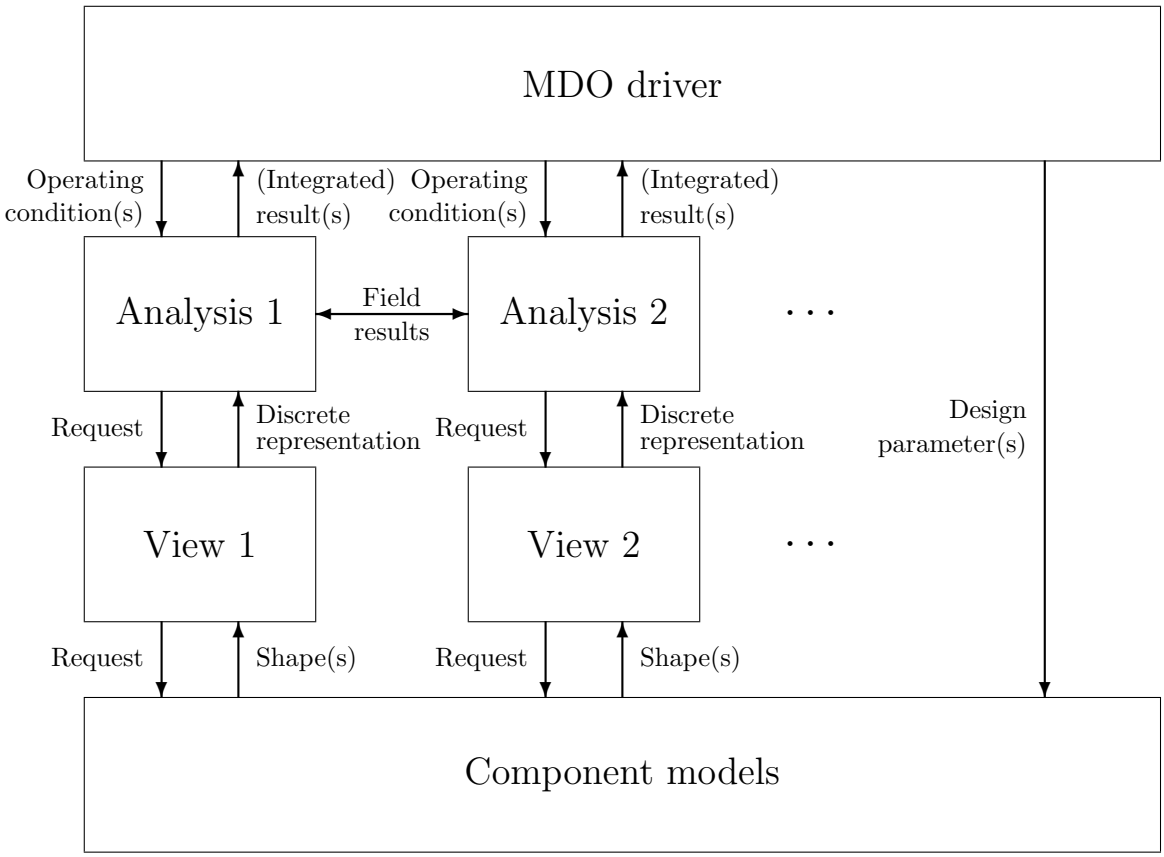
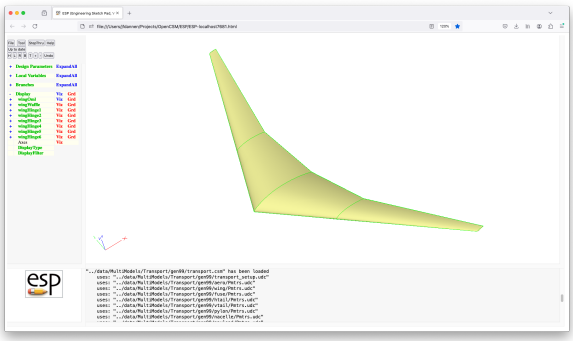
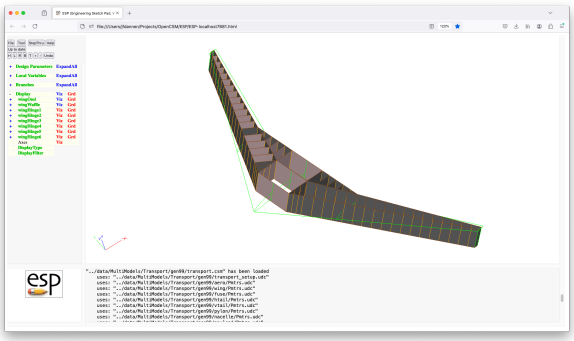


Figure 1. Overall system organization. All transfers include both value and sensitivities

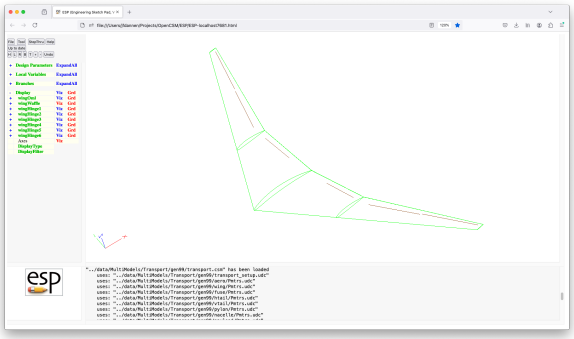
Notice that none of these component models represent a final geometry, but instead are the building blocks from which the various "views" are constructed.



(a) (solid) OML



(b) waffle (for structures)



(c) hinge lines

Figure 2. Component models for transport wing. Note that the outline of the OML is shown in green in all figures to show the relative locations of the various component models.

The common set of parameters that drives these models are shown in Table. 1. Note that the planform parameters (`wing:aspect`, `wing:area`, `wing:sweep`, ...) are useful for not only describing the outer mold line but also essential for the placement of structural elements, such as the wing spars.

During the creation of these component models, it has been found to be very beneficial to define attributes on the various parts of each view; this can be done with a set of "tags". For the wing structure, appropriate tags that are easy to apply when the component model is first generated might be:

- **tagComponent**: the name of the component such at `leftWing`;
- **tagType**: the type of surface, such as `rib` or `spar`;
- **tagIndex**: the index of the surface, such as `2` for the second rib; and
- **tagSubIndex**: (where appropriate) the sub-part of the surface, such as `1` for the first part of a rib (that is, the part of the rib between the leading edge and the first spar).

In theory is should be possible to ascertain these attributes by reverse-engineering the final model(s), but practice has shown that such a process is often difficult and prone to errors.

Note that some of the component models are notional. For example, the component model for the various control surfaces on a wing (ailerons and flaps) shown in Fig. 2(c) are described simply in terms of their hinge line, with other associated data such as gaps and deflections prescribed via attributes on the hinge lines.

Table 1. Wing design parameters.

```
# wing Oml
DESPMTR   wing:area        4240    # area
DESPMTR   wing:aspect      9.00    # aspect ratio
DESPMTR   wing:taperi      0.48    # inboard  taper ratio
DESPMTR   wing:tapero      0.23    # outboard taper ratio
DESPMTR   wing:sweep       35.0    # leading edge sweep
DESPMTR   wing:dihedral     7.0    # dihedral
DESPMTR   wing:break       0.37    # inboard/outboard
DESPMTR   wing:alphar      -1.0    # setting angle   at root
DESPMTR   wing:thickr      0.10    # thickness ratio at root
DESPMTR   wing:camberr     0.08    # camber    ratio at root
DESPMTR   wing:alphab      -3.0    # setting angle   at break
DESPMTR   wing:thickb      0.15    # thickness ratio at break
DESPMTR   wing:camberb     0.04    # camber    ratio at break
DESPMTR   wing:alphat      -8.0    # setting angle   at tip
DESPMTR   wing:thickt      0.08    # thickness ratio at tip
DESPMTR   wing:cambert     0.01    # camber    ratio at tip
DESPMTR   wing:xroot       50.0    # xloc at root LE
DESPMTR   wing:zroot       -8.0    # zloc at root LE
DESPMTR   wing:tipTreat     2.0    # tip treatment (or 0 for none)
DESPMTR   wing:wakelen     10.0    # number of wing:mac from TE

# wing/fuselage fillet
DESPMTR   wing:fillet       1.0    # fillet radius at fuselage

# wing hinge lines
DIMENSION wing:hinge        6 9 1 # ymin                  ymax
#                           theta   x/c   y/span  z/t     x/c   y/span z/t    gap   grp
DESPMTR   wing:hinge       "-10.0;  0.75; -0.98;  0.50;   0.75; -0.70; 0.50;  0.25; 1;  \ left aileron
                            +10.0;  0.75; -0.69;  0.00;   0.75; -0.43; 0.00;  0.25; 2;  \ left oflap
                            +15.0;  0.85; -0.33;  0.00;   0.90; -0.14; 0.00;  0.25; 3;  \ left iflap
                            +15.0;  0.90;  0.14;  0.00;   0.85;  0.33; 0.00;  0.25; 3;  \ rite iflap
                            +10.0;  0.75;  0.43;  0.00;   0.75;  0.69; 0.00;  0.25; 2;  \ rite oflap
                            +10.0;  0.75;  0.70;  0.50;   0.75;  0.98; 0.50;  0.25; 4"  # rite aileron

# wing structure
DESPMTR   wing:spar1       0.20    # fraction of chord for LE spar
DESPMTR   wing:spar2       0.70    # fraction of chord for TE spar
CFGPMTR   wing:nrib1          2    # number of internal ribs in region 1
CFGPMTR   wing:nrib2          4    # number of internal ribs in region 2
CFGPMTR   wing:nrib3         12    # number of internal ribs in region 3

DESPMTR   wing:waffleGap      1    # distance between fuselage and wing root rib
```

D. Views

Above the component models in Fig. 1 are the **Views**, which are simply a representation of the geometry (and perhaps the associated mesh) that are required for the various analyses. The views of the wing that can be simply be derived from the wing's component models are shown in Fig. 3.

Fig. 3(a) shows the geometric representation of the wing that is required for the AVL[4] vortex-lattice method. `AVL` requires that the wing be described in terms of streamwise cross-sections at all places where the spanwise geometry changes (such as at the root, wing break, and tip) as well as at the inboard- and outboard-extents of the various control surfaces. Hence the Vlm view of the transport wing consists of 17 cross-sections, all flying in formation.

Fig 3(b) shows the geometric representation of the wing that might be appropriate for a panel method. For this analysis the control surface are not actually cut into the wing, but are instead outlined so that the panel method can simply "rotate the normals" to model various control surface deflections.

The control surfaces are treated very differently for CFD calculations, depending on the physical model. For example for an inviscid calculation, the flow of air between the main wing and the control surface is generally not modeled since it would incur a large computational expense for a region of the domain that would not be modeled properly anyway. So for the inviscid CFD analysis a gap-filler is applied, as shown in Fig. 3(c). On the other hand, when computing the viscous flow, the flow through the gap is important, and so the gap is correctly modeled, yielding several bodies flying-in-formation, as shown in Fig. 3(d).

The most complex view for the wing is actually the built-up element model, as shown in Fig. 3(e) and 3(f), with and without skins respectively. This model is a series of shell elements that are constructed from the OML and a "waffle" by the following steps:

- the waffle and the (solid) OML are intersected, yielding the part of the waffle inside the wing (that is, the ribs and spars);
- the skins of the OML are extracted and are scribed by the waffle, thereby separating the wing skin into the panels between the spars and ribs;
- these two models are combined into a single sheet-body that represents the spars, ribs, and skin panels; and
- if only the wing-box is required, the skin panels that are adjacent to the leading- and trailing-edges are removed.

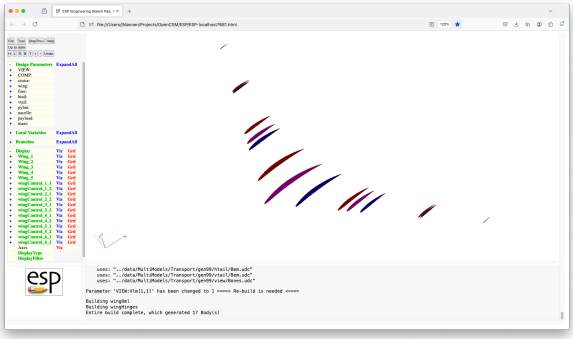
This latter step uses the attributes on the OML component model that identified the leading- and trailing-edges during the construction process.

E. Analyses

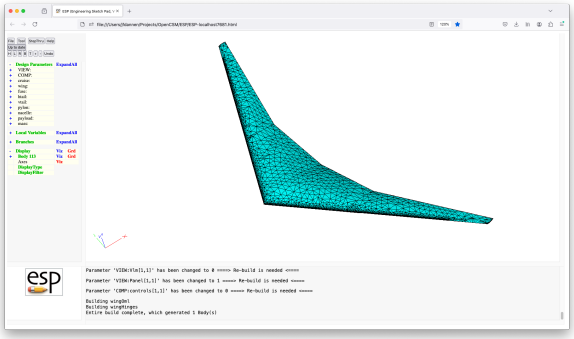
Associated with each view is an **analysis** method, which can largely be treated as a black box. These analysis methods can be quite expensive to execute, often requiring that they be distributed in a high-performance computing environment; the distribution throughout the network is one of the strengths of many modern MDO frameworks.

F. MDO framework

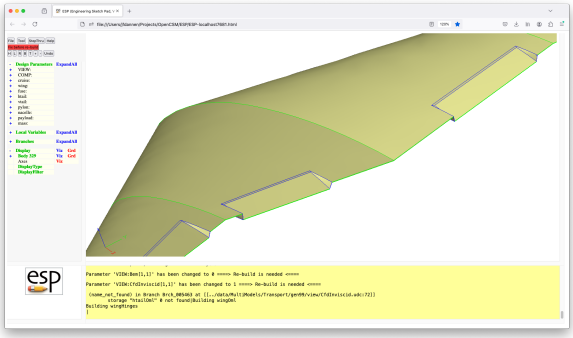
At the top of Fig. 1 is the MDO framework which drives the entire MDO process. It is responsible for defining the latest set of design parameters (which drive the various component models and hence views) and for scheduling the execution of the various analyses. Any of the currently-available MDO frameworks, such as ModeFRONTIER, ModelCenter, or OpenMDAO can be used for this.



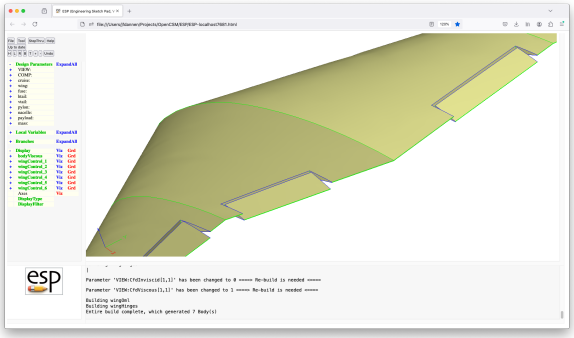
(a) vortex-lattice cross-sections



(b) panel

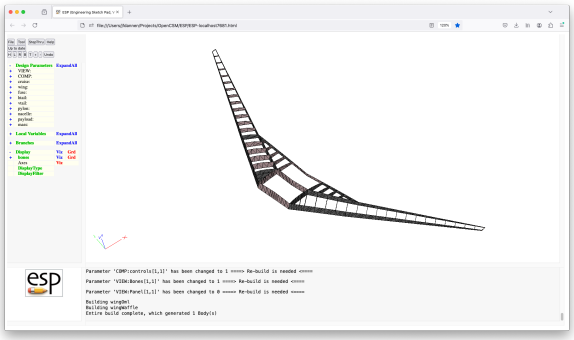


(c) CFD inviscid
(looking forward)



(d) CFD viscous
(looking forward)

(e) built-up element method
(ribs, spars, and skins)



(f) built-up element method
(ribs and spars only)

Figure 3. Views for transport wing

G. Communication

The communication between the various parts of the system is described next. In a typical step of the optimization process, the following are performed:

- the MDO framework sets values for the various design parameters, It may also define "design velocities", which are a way of specifying the required sensitivities;
- the MDO framework executes the needed analysis by:
  - setting the operating conditions (such a Mach number and angle of attack) and possibly their velocities;
  - calling the analysis;
  - in turn, the analysis may request that the appropriate view be generated;
  - in turn, the needed component models are generated. In addition to shape, the component models may include sensitivities (that are derived from the specified design parameter velocities);
  - the view is then assembled, which is comprised of a discrete representation of the associated geometry as well as the sensitivity of that discrete representation with respect to the design parameter velocities;
  - the analysis is executed. Note that the analysis may share "field results" with other analyses. For example, an aerodynamic analysis may depend on surface deflections that are generated by a structural analysis. Similarly, the aerodynamic analysis may share its surface pressure distribution with a structural analysis in order to prescribe the structural loads. These transfers can either be done via interpolation or may be done "conservatively" (that is be transferred such that the integrated loads in the two models match);[5] and
  - the analysis transfers its results (and possible sensitivities) back to the MDO framework. Note that the MDO frameworks typically require integrated results, such as lift and drag coefficients of factors of safety.
- The MDO framework combines the results from the various analysis in order to make the next optimization step.

III. Implementation in the Engineering Sketch Pad (ESP)

The Engineering Sketch Pad (`ESP`)[6] is a geometry creation and manipulation system whose goal is to support the analysis methods used during the MDO design process. It is a solid modeler that guarantees that models are realizable solids, with a watertight representation that is essential for mesh generators. `ESP`'s models are parametric, meaning that they are defined in terms of a feature tree (which can be thought of as the "recipe" for how to construct the configuration) and a set of user-defined design parameters (DESPMTRs) that can be modified to generate families of designs. `ESP` maintains a set of global and local attributes on a configuration that are persistent through rebuilds, which is essential in the support of multi-fidelity models (wherein the attributes can be used to associate conceptually-similar parts in the various models) and multi-disciplinary models (wherein the attributes can be used to associate surface groups which share common loads and displacements). A key difference from `ESP` and all other available modeling systems is the `ESP` allows a user to compute the sensitivity of any part of a configuration with respect to any design parameter. `ESP`'s user interface runs in any modern web browser and its calculations are executed in a server-based backend program, making it ideally suitable for use by an MDO framework. As part of the `ESP` distribution is the Computational Analysis Prototype Synthesis (`CAPS`)[7] program that is the linkage between the geometry system and over 30 analyses. `ESP` (and `CAPS`) are open-source projects (using the LGPL 2.1 license) that are distributed as source, and are available from `acdl.mit.edu/ESP`.

The techniques described above have been applied to a transport configuration. The biggest change is the size of the model; the final model is comprised of 53 files, that contain over 3800 lines of `.csm` code. Also,

the full transport is defined in terms of about 100 design parameters and about 20 configuration parameters (which are used to turn various parts of the configuration off and on).

Fig. 4(a) shows an overall concept view, which is generally used by the designer to ensure that the various design parameters "make sense". For example, by examining the concept view, the user can see the relative placement of the various internal components; also the user can ensure that the control surface hinge lines lie outside and parallel to the wing box.

Fig. 4(b) shows the vortex-lattice view for the entire configuration and Fig. 4(c) shows the panel view. Here some of the major components are color coded. This view is constructed by "unioning" the OMLs of the various components. Fig. 4(d) shows a portion of the viscous CFD view; note that here the gap between the main wing and the control surfaces is modeled directly, yielding several bodies flying in formation.

The creation of the structure was quite a bit more complex than the aerodynamic models. Recall that the aerodynamic models were created by "unioning" the various component models: the wing was combined with the fuselage, to which the horizontal tail was combined, to which the vertical tail was combined, ... In this case, each of the component models could be generated in isolation.

But for the structure, there was an interaction between the components that had to be considered. For example, in order to lay out the wing structure, one needs to know the fuselage width so that appropriate structural elements can be generated at the wing/fuselage junction. Similarly, there needed to be bulkheads in the fuselage that align with the wing spars; the remaining fuselage bulkheads were placed so as to evenly divide the portion of the fuselage ahead of the wing and between the wing and tail. It took several attempts at creating the structures model until all the interactions were properly handled.

IV. Summary

In order for geometry to play a central role in MDO processes, it must:

- provide the necessary multi-fidelity and multi-disciplinary "views" of the configuration for the associated analyses;
- drive the generation of the views from a shared set of design parameters (that can be adjusted by the MDO framework);
- provide rich attribution so that parts of the configuration in the various views can effectively communicate with each other; and
- provide sensitivities of the configuration with respect to any desired set of design parameters (if a gradient-based optimizer is to be used).

A system architecture that satisfied these requirements is described and an implementation of it using the Engineering Sketch Pad (`ESP`) is shown. The design of a transport-type configuration is used as an example.

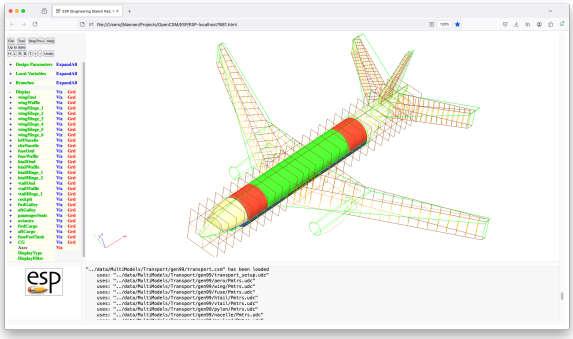
Acknowledgment

This work was supported by the EnCAPS Project (AFRL Contract FA8650-20-2-2002): "EnCAPS: Enhanced Computational Prototype Syntheses", with Joshua Deslich and Joshua Deaton as the Technical Monitors. The author would also like to thank Bob Haimes and Nitin Bhagat for all their useful insights during the development of this process.

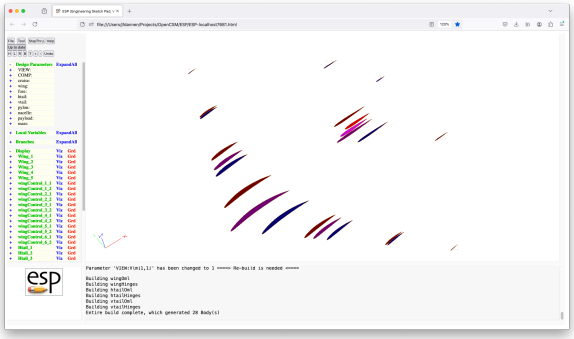
References

[1]Nardin, L., Sorensen, K., Hitzel, S., and Tremel, U., "modeFRONTIER, a Framework for the Optimization of Military Aircraft Configurations", Notes on Numerical Fluid Mechanics and Multidisciplinary Design, (NMFM, volume 107), Springer, 2009.

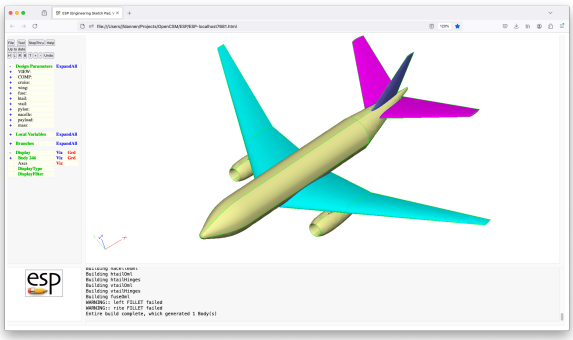
[2]`https://www.ansys.com/products/connect/ansys-modelcenter`



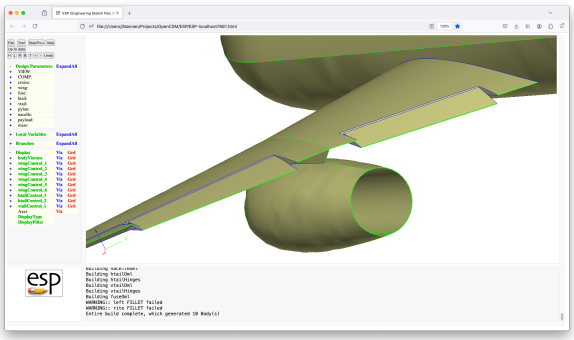
(a) concept



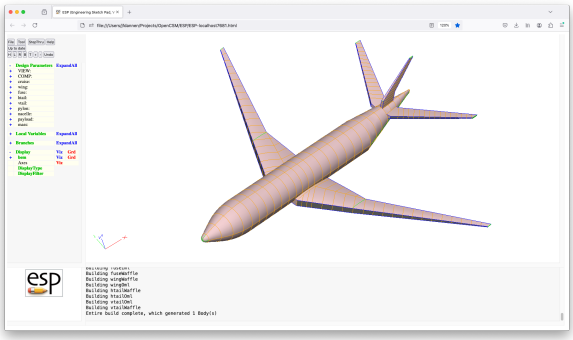
(b) vortex-lattice cross-sections



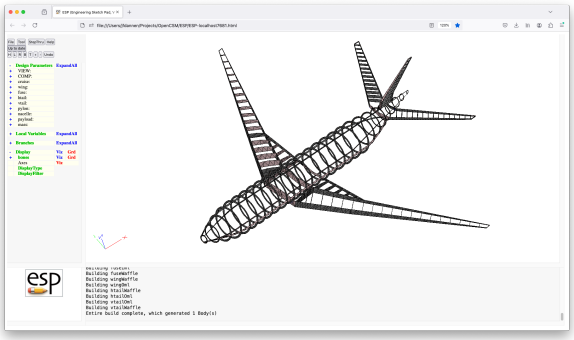
(c) panel



(d) CFD viscous
(looking forward)



(e) built-up element method
(ribs, spars, and skins)



(f) built-up element method
(ribs and spars only)

Figure 4. Views for transport

[3]Gray, J.S., Hwang, J.T., Martins, J.R.R.A, Moore, K.T., and Naylor, B.A., "OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization", *Structural and Multidisciplinary Optimization*, pp 1075-1104, April 2019.

[4]Drela, M., and Youngren, H., "AVL User Primer", `https://web.mit.edu/drela/Public/web/avl/avl_doc.txt`, Feb 2022.

[5]Dannenhoffer, J.F., and Haimes, R., "Conservative Fitting for Multi-Disciplinary Analysis", AIAA-2014-0294, January 2014.

[6]Haimes, R. and Dannenhoffer, J.F., "The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry", AIAA-2013-3073, June 2013.

[7]Bryson, D.E., Haimes, R., and Dannenhoffer, J.F., "Toward the Realization of a Highly Integrated, Multidisciplinary, Multifidelity Design Environment", AIAA-2019-2225, January 2019.