AIAA AVIATION Forum
June 27-July 1, 2022, Chicago, IL & Virtual
AIAA AVIATION 2022 Forum

10.2514/6.2022-3493

GLOVES: A Graphical Layout of Vehicle Systems for the Engineering Sketch Pad

Paul R. Mokotoff * and John F. Dannenhoffer, III †
Aerospace Computational Methods Laboratory, Syracuse University, Syracuse, NY, 13244

During conceptual aircraft design, the layout (arrangement and size) of an aircraft's components are continuously updated. Current computational environments, such as NASA's Open Vehicle Sketch Pad, allow for the designer to indirectly manipulate the layout by using sliders in a separate window. However, there is no way for the designer to directly interact with the components and change their shape or size. Furthermore, the user is not given any immediate feedback on how the design change affected the overall system.

The Graphical Layout of Vehicle Systems, GLOVES, allows a designer to interactively arrange and size predefined or custom aircraft components within the Engineering Sketch Pad software. That way, the designer can create an aircraft layout starting with the payload arrangement and finish with a completed outer mold line. The distinguishing features of GLOVES include: (1) the ability to directly manipulate the geometry in real-time; (2) a weight and balance analysis that updates both the components' and overall centers of gravity as the geometry changes. These features provide the user with immediate feedback on how changing the geometry affects the layout and its mass properties. To demonstrate the current capabilities of GLOVES, a simple transport aircraft layout is created. It is shown that interactively manipulating the components in a graphical user interface is more effective than using sliders.

I. Nomenclature

CAPS	=	Computational Aircraft Prototype Synthesis
CG	=	Center of Gravity
ESP	=	Engineering Sketch Pad
GLOVES	=	Graphical Layout of Vehicle Systems
GUI	=	Graphical User Interface
OpenVSP	=	Open Vehicle Sketch Pad
TIM	=	Tool-Interface Module
UDP	=	User-Defined Primitive
VBO	=	Vertex Buffer Object
2D	=	Two-Dimensional
3D	=	Three-Dimensional

II. Introduction

During conceptual design, a preliminary layout of the aircraft and its major components (payload, fuselage, wing, tail, etc.) is developed. The designer explores multiple aircraft layouts and determines the best one that satisfies the mission - both performance and payload - criteria. After drawing rough sketches, simple geometric models are created in a computer-aided design system. Once these models are made, the designer will interact with the layout to improve it or explore alternative ones. Current computational environments for aircraft modeling and design, such as NASA's Open Vehicle Sketch Pad (OpenVSP), only allow the designer to interact with the layout indirectly via sliders [1]. Figure 1 illustrates two of the windows that an OpenVSP user interacts with: (1) the main workspace (left), used for transforming the view of the layout, and (2) the geometry window (right), used for changing the design parameters governing the

*Graduate Student, Department of Mechanical & Aerospace Engineering, 263 Link Hall, Syracuse, NY 13244, AIAA Student Member.
†Associate Professor, Department of Mechanical & Aerospace Engineering, 263 Link Hall, Syracuse, NY 13244, AIAA Associate Fellow.

layout. As the user adjusts one of the sliders in the geometry window, the layout in the main workspace is re-drawn and the parameters dependent on the one being changed are updated.

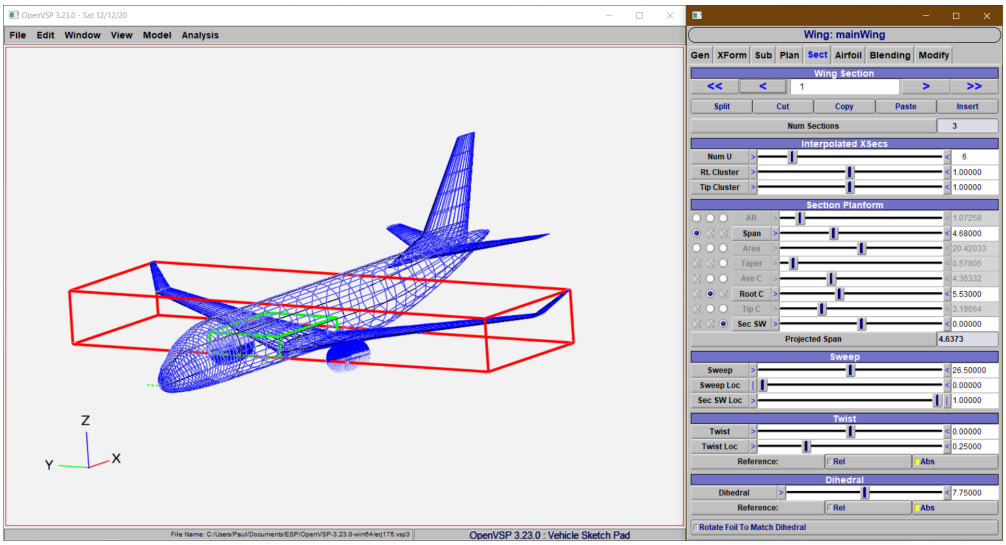


Fig. 1 OpenVSP's main workspace (left) and geometry manipulation window (right) [2].

One advantage of using OpenVSP is that the user can add custom components to their design by writing a VSPPART file [2]. The geometry of the new components and the sliders needed to modify it can be defined in a maximum of six functions [3]. This allows the user to include proprietary or unique components in the layout. A second advantage of OpenVSP is that the geometry can be connected to a variety of analysis tools. In OpenVSP, there are geometric analysis tools that allow a user to compute the mass properties or projected area in a given direction, to name a couple. There are also tools for low-fidelity aerodynamic analysis such as VSPAERO, a potential flow solver [3], and the Parasite Drag and Wave Drag tools [2]. All of these tools use a graphical user interface (GUI) similar to the one used for geometry manipulation.

One disadvantage of using OpenVSP is that the software is more suited for modeling aircraft rather than designing aircraft. If the mission criteria were to change after the aircraft layout was already made, significant time and effort would be required to modify the existing design. Each component needs to be individually adjusted by the sliders and for designs with more than 5 components, the layout modification becomes a tedious process. Additionally, not all design parameters are listed on one menu. For example, the position of the wing is listed on a different menu than the nominal wing dimensions. This requires the user to navigate between menus and find the desired design parameter while manipulating a component. It would be simpler if the designer knew the design parameters that affected the geometry the most, and directly interact with the components on the main workspace by manipulating them with their mouse. Another disadvantage of using OpenVSP is that the computed mass properties are only valid for the existing configuration, must be re-computed upon manipulating the geometry, and cannot be updated in real-time. It would be more instructive if the designer received immediate feedback and knew how manipulating the layout affected its mass properties.

In addition to OpenVSP, the Engineering Sketch Pad (ESP) currently has a limited version of GLOVES, which allows users to layout wing, fuselage, and tail components on a screen and directly manipulate a wireframe representation in the GUI (Figure 2) [4]. The components in GLOVES are drawn using a two-dimensional (2D) canvas rendering context in a web-based browser and are wireframes comprised of only line segments [4]. On each component, there are small pips (points) at its vertices. As the user moves the mouse and hovers over each pip, a menu is displayed on the screen and lists the available design parameters that can be changed, as depicted in Figure 3a. After selecting a design parameter to be changed, the user moves their mouse on the screen to directly manipulate the wireframe. As shown in

Figure 3b), the pip that was selected turns red and the design parameters governing the shape and size of the component are listed in the upper left corner of the canvas. After the desired configuration is attained, the user left-clicks to stop manipulating the wireframe, and the layout is re-built.

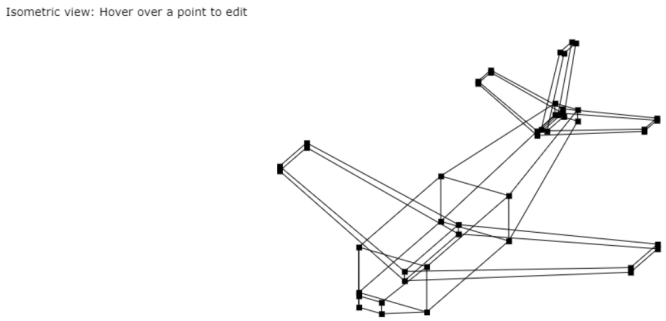
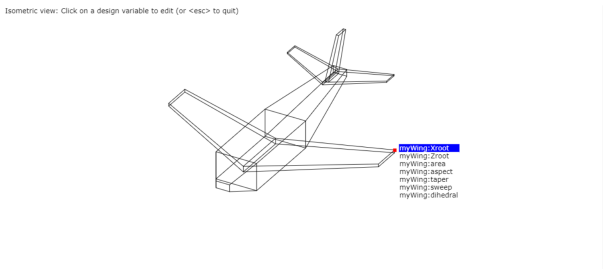
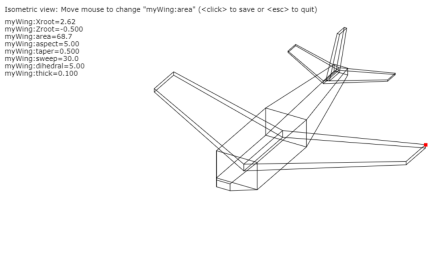


Fig. 2 Limited version of GLOVES that already exists in ESP [4].

An advantage of using the current implementation of GLOVES is that the user can easily modify the layout if the mission were to change. This is done by hovering over the pips on each component and manipulating the geometry, thus changing the design parameter values. The process to update the design will be faster in ESP than OpenVSP because the user does not waste time selecting each component individually, navigating to, and moving multiple sliders - the user can hover over the component and change it immediately. Another advantage of GLOVES is that the user can connect the geometry to both low- and high-fidelity analysis tools via CAPS, an Application Programming Interface that transforms the geometry in ESP into an input file for one of the analysis tools. However, this is currently a moot point because the configurations developed in GLOVES are quite simplistic and need to be refined outside of GLOVES before undergoing any analyses.



(a) Menu available to manipulate the geometry.



(b) Manipulating the area of the wing.

Fig. 3 Example operations in the limited version of GLOVES.

There are some disadvantages to the current implementation of GLOVES in ESP. First, all of the current GLOVES operations are done by manipulating a wireframe. Although the user interacts with the wireframe directly, there is no interaction with the final geometry. Upon saving and exiting GLOVES, the geometry is re-built, and will not exactly match the user's design in the tool, as seen in Figure 4. Second, only three components are available: (1) a wing/horizontal tail; (2) a fuselage; and (3) a vertical tail. Currently, the user cannot add custom components, making this version of GLOVES extremely limited in its capabilities. In order to develop more complex configurations, other components such as different shaped payloads, control surfaces, internal structures, and landing gear must be available.

Lastly, the user is only limited to four views: (1) an isometric view; (2) a side view; (3) a front view; and (4) a top view. Since the components are drawn on a 2D canvas, it is sometimes difficult for the user to gauge depth, especially in the side, front, and top views. As a result of this, the user is not always able to effectively interact with the wireframe.

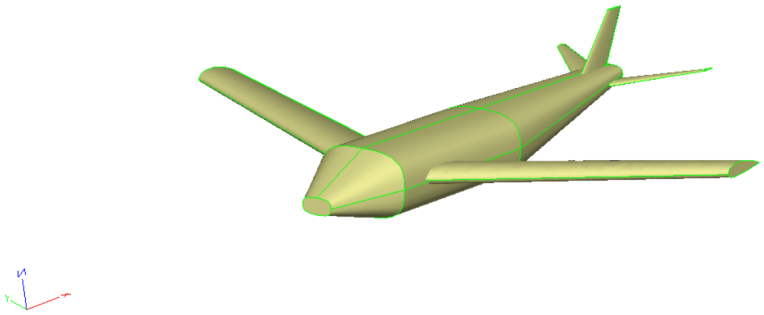


Fig. 4 GLOVES configuration after building the geometry.

III. Objective

The objective of this research is to develop a new version of GLOVES that allows a user to design an aircraft graphically and modify the geometry directly in the ESP GUI. That way, the user can create an aircraft layout starting with the payload arrangement and finishing with a completed outer mold line. In order for the user to interact with the geometry directly, sensitivities are used. While manipulating the geometry, the layout only needs to be re-drawn on the canvas. Once the user is done manipulating the geometry, it is then be re-built and re-drawn. As a result, the proposed geometry manipulation process will be less computationally expensive and time consuming than repeatedly re-building the geometry for every mouse movement made by the user.

Rather than creating 2D wireframes, three-dimensional (3D) solids are created via User-Defined Primitives (UDPs) specifically for GLOVES*. This gives the user the freedom to add their own custom components and makes it easier to include newly developed components into future iterations of GLOVES. Since the UDPs create primitive solids, GLOVES is housed in the same 3D canvas rendering context as the ESP graphics window. The user can easily pan, zoom, or rotate the view, thus making it easier to interact with the geometry.

In addition to its geometry creation and manipulation capabilities, GLOVES has a feature that allows the designer to visualize the center of gravity of each component as well as that of the entire layout. By using sensitivities, the mass properties are updated in real-time, thus allowing the user to understand how the proposed modification affects the weight and balance of the overall design. This feature is particularly useful for modifying existing aircraft designs. For example, if the fuselage is lengthened and engines are enlarged on a layout, the designer knows how much the center of gravity has shifted from its original location.

IV. GLOVES Operations

A. Launching GLOVES

To initialize a session of GLOVES, ESP must be launched first. After ESP has started, the user left-clicks the "Tool" button and selects "Gloves" from the "Tool Menu", as shown in Figure 5. Figure 6 depicts what GLOVES looks like

*A UDP is a piece of C, C++, or FORTRAN code written by a user to create a primitive solid that doesn't already exist in ESP [5].

upon initializing a new session with multiple components in the CSM script. If no components were given in the CSM script, the canvas only contains the axes in the bottom left corner.

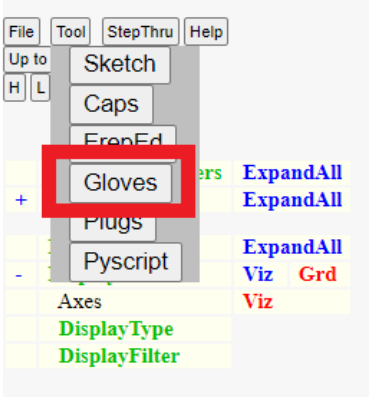


Fig. 5 Launching GLOVES from the "Tool Menu".

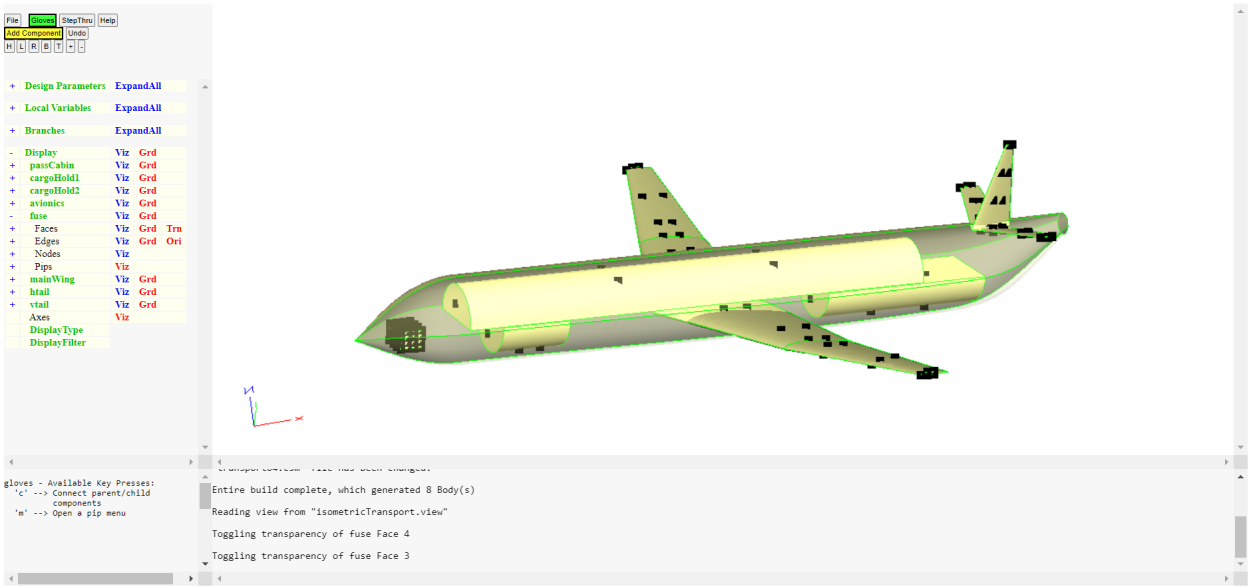


Fig. 6 GLOVES after a new session has been started.

In the top left pane, the Tree Window, there are two buttons that a GLOVES user selects. In the top left corner, a button labeled "Gloves" appears in green. After clicking on this button, a dropdown menu contains a list of the available analyses that can be performed on the layout, and also allows the user to either save and exit the session or quit without saving. Currently, there is only one analysis option, "Toggle CGs", which allows the user to see the CG of each component and that of the entire layout. Below the "Gloves" button is another yellow button labeled "Add Component". After clicking the button, the user is prompted with a list of components from the two available systems: (1) the Payload System, and (2) the Aerodynamic System. (The components in these systems will be presented in subsection V.A.) The user is also given an option to add a custom component to the layout.

Below the buttons is a tree of design parameters, local variables, branches, and displays. These trees are filled as a user adds components to the layout. In the "Design Parameters" portion of the tree, the user can view and modify the

value of a design parameter by clicking on its name (in addition to hovering over a pip and modifying the geometry directly). This is particularly useful for making changes to a vector valued design parameter, such as the camber or thickness of multiple airfoils in a wing or tail. In the "Local Variables" portion of the tree, the user can view two mass properties for each component: (1) its volume and (2) center of gravity. In the "Branches" portion of the tree, the user can see a list of the branches used to generate the layout. However, the user cannot modify the branches directly in GLOVES and must wait to do so until saving or quitting. Lastly, the "Display" portion of the tree allows the user to change the visibility of the graphics primitives on the 3D canvas. The user can also modify the transparency of the faces on each component, which is particularly useful when viewing the CGs.

The top right pane, also known as the Graphics Window, contains two canvases. One of them contains a 3D rendering context, which is used for displaying the layout and the axes drawn in the bottom left corner of the Graphics Window. The second canvas contains a 2D rendering context and displays text. When hovering over a component and selecting a pip, the 2D canvas is placed in-front of the 3D canvas, thus allowing the user to select a design parameter to modify. When manipulating a component, the 2D canvas displays the proposed value of the design parameter being modified.

The bottom left pane, also known as the Key Window, lists the available key presses that a user can perform to interact with the layout. The Key Window is updated when a user selects a pip, actively modifies a design parameter, or links multiple components together. Lastly, the bottom right pane, also known as the Message Window, will occasionally display a warning or notification to the user. Messages are typically posted when the user makes an error such as selecting an invalid value for a design parameter or linking two components that are already connected.

B. Layout Arrangement Workflow

The goal of GLOVES is to allow the user to build up an aircraft layout from the payload to the outer mold line. After arranging the payload, the user can surround it with a fuselage. Once the fuselage is placed, the user adds a wing to the layout. By using the weight and balance analysis tool, the wing can be placed such that the center of gravity of the layout aligns with the quarter-chord. Upon placing the wing, the user can add and size the horizontal and vertical tail components based on the size and position of the wing.

To perform this workflow in GLOVES, the user first adds a component to the layout. This is done by selecting the "Add Component" button, and selecting a component from the Payload System. After the user selects which component to create, a 3D solid body is created. At the same time, the sensitivities of the component with respect to its design parameters are computed. Once the sensitivities have been computed, the pips and their menus are created. The menu on each pip contains a ranked list of the design parameters that affect the configuration the most at that point. After the menus are created, the solid body and pips are drawn on the canvas, and the user can begin manipulating the geometry.

To manipulate the geometry, the user first selects a pip and a design parameter from its menu. At this point, the user moves the mouse on the canvas and the pip begins to follow the mouse movements. This is done internally by choosing a step size that minimizes the distance between the pip on the canvas and the mouse position. Once this step size is chosen, the VBOs are re-drawn and the geometry is deformed. Once the user has finished manipulating the geometry, they left-click on the final pip position. After this, the geometry is re-built and the sensitivities are computed again, thus creating new menus on each of the pips. The user will continue manipulating the geometry until the desired shape and size is attained. Once the desired geometry is made, a new component is added to the system and the cycle repeats itself. A diagram of the overall workflow is illustrated in Figure 7.

V. GLOVES Features

A. Predefined and Custom Components

In GLOVES, there are currently six predefined components that a user can add to the layout. Each component is created based on a set of design parameters and (possibly) configuration parameters[†]. The components in GLOVES are currently grouped into two 'systems': (1) the Payload System and (2) the Aerodynamic System.

[†]A design parameter is a value whose sensitivities can be computed. A configuration parameter is a value whose sensitivities cannot be computed [6].

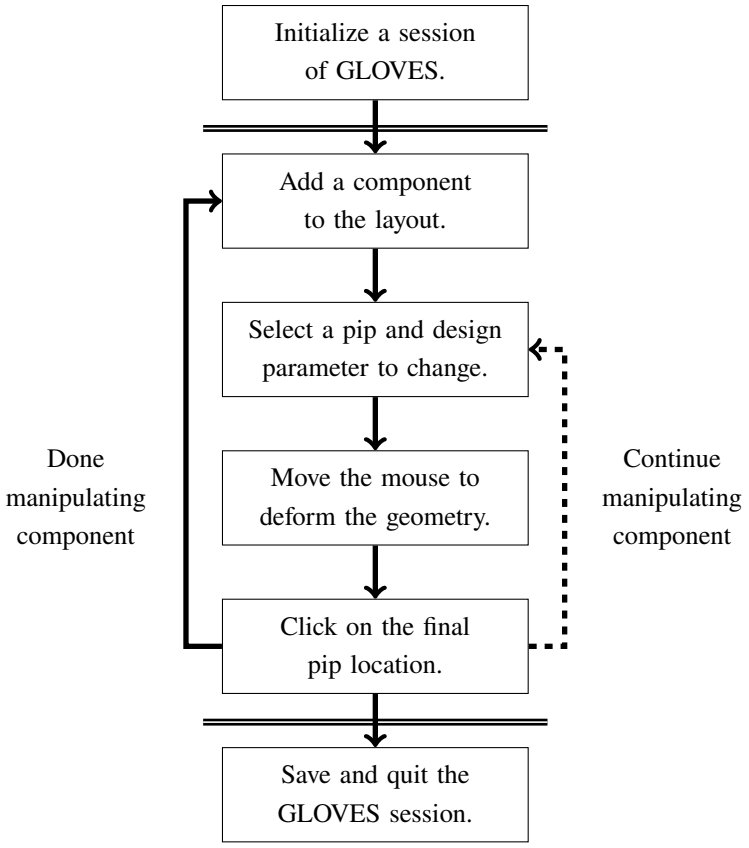


Fig. 7 Workflow in GLOVES for adding and manipulating components.

From the Payload System, the user can add a: (1) box, `glovesBox`, (2) cylinder, `glovesCyl`, or (3) sphere, `glovesSphr`. If a user adds a cylinder, they have the option to build either the top ($+z$), bottom ($-z$), or both halves of the cylinder. Selecting to build only the top half of the cylinder is useful for creating a cabin for the passengers and galleys. Selecting to build only the bottom half of the cylinder is useful for creating a cargo bay or fuel tank in the fuselage.

From the Aerodynamic System, the user can add a:

1) Fuselage, `glovesFuse`, which is made of a set of blended superellipses. On each superellipse, the user can control the position, nominal radii in the y- and z-directions, and its power. The user is also given control over the continuity in the blend direction and can select: C2 continuity, curvature continuous; C1 continuity, slope continuous; or C0 continuity, value continuous [7]. For this component, the default continuity at each superellipse is set to be C2 continuous. However, the user may want to modify the continuity to be C1, particularly for the portion of the fuselage that contains the passenger cabin.
2) Symmetric wing or horizontal tail, `glovesWing`, which is made of a set of blended NACA airfoils along the y-axis. On each airfoil, the user can control the camber, thickness, location of maximum thickness, angle of attack, and position as a function of the wingspan. On each section of the wing (the area bounded by two consecutive airfoils), the user can control the taper ratio, sweep angle, dihedral. Just like the fuselage component, the user is also given control over the continuity in the blend direction. For the wing, the default continuity at each airfoil is set to be C0 continuous, thus creating a ruled wing. The user may want to modify the continuity towards the outboard portions of the wing to create a blended wingtip.
3) Vertical tail, `glovesVtail`, which is also made of a set of blended NACA airfoils. The user is given the exact same control over this component as they are with the wing. The only difference between the wing and vertical tail components is that the blend direction for the vertical tail is parallel to the $+z$ axis (as opposed to the $+y$-axis).

In addition to the predefined components, the user can elect to create their own custom components and include them in a layout. In order for a custom component to be created, the user must write a UDP, which contains a series of function calls to `EGADS` [5]. At a high-level, these function calls are used to create the component's topological entities (faces, edges, and nodes). Additionally, for GLOVES components, the user must specify the location of the pips on each face and whether any 'special' menu options should be displayed when a pip is selected. The location of the pips on each face is specified by adding an attribute, `_.pips._` to the face. The attribute contains an array of doubles consisting of a u-v coordinate pair for each pip. If no pips are to be placed on the face, the attribute should be assigned a string value of `NULL`.

'Special' menu options may be required for some pips. Typically, the menu on a pip only contains a list of the design parameters with the largest sensitivities. However, there may be times in which a user wants to change a configuration parameter, or modify the number of sections in a blend. If the user feels that this is necessary for their custom component, the `_.type._` attribute needs filled. The attribute consists of a pair of numbers for each pip. The first number is a bitwise representation of the menu options to be added at the pip. The second number in the pair is the index of the section that the menu option pertains to, or -1 if it is not applicable (like changing the half of a cylinder to build). The options for the first number of the pair include:

1) `GLOV_HERE`: used on pips between two sections. This allows a user to add a section halfway between the two existing sections.
2) `GLOV_BEFR`: used on pips at a section. This allows a user to add a section before the index indicated on the menu option.
3) `GLOV_AFTR`: used on pips at a section. This allows a user to add a section after the index indicated on the menu option.
4) `GLOV_DLTE`: used on pips at a section. This allows a user to delete the section given by the index on the menu option.
5) `GLOV_CURV`: used on pips at a section. This allows a user to change the blend continuity given by the index on the menu option.
6) `GLOV_SECT`: used on pips at the beginning or end of a blended body. This allows a user to add another section before the first section or after the last section.
7) `GLOV_NODE`: used on pips at the beginning or end of a blended body. This allows a user to add a node (to blend to a point) before the first section or after the last section.
8) `GLOV_ARFL`: used on pips at a section. This allows a user to specify the NACA 4-digit series airfoil at a section.
9) `GLOV_HALF`: used on pips for a cylinder. This allows a user to specify whether the top ($+z$), bottom ($-z$), or both halves of the cylinder are built.

The user can include an additional bitwise representation to create a new menu option, but it may be overwritten in future iterations of GLOVES. The bitwise representations of the menu options can be found in `gloves.h` that is shipped with ESP.

Aside from creating the UDP, the user must also update two functions in the source code, `gloves.c`, that is shipped with ESP. The first function is `compInit`, which gets the default design/configuration parameter values when the component is added. Here, the user must fill the following output variables:

1) `ptype`: the parameter type for ESP to recognize. This will either be `OCSM_DESPMTR` for a design parameter or `OCSM_CFGPMTR` for a configuration parameter.
2) `gtype`: the parameter type for GLOVES to recognize. This will either be `GLOV_SCALAR` if the parameter is scalar valued or `GLOV_VECTOR` if the parameter is vector valued.
3) `nvalu`: the number of values associated with the default parameter. This is either 1 for a scalar valued parameter or the length of the vector for a vector valued parameter.
4) `valu`: an array of the default value(s).
5) `lbnd`: an array of the lower bound(s). If there is no lower bound, use an arbitrarily large number like -10^{10}.
6) `ubnd`: an array of the upper bound(s). If there is no upper bound, use an arbitrarily large number like $+10^{10}$.

The second function to be updated is `compSect`, which computes the new design/configuration parameters for a component when a section is added or removed. For example, if a section is added on the `glovesWing` component, a

new taper ratio, sweep angle, and dihedral must be computed before building the modified component. There are no particular rules associated with modifying this function. If the component does not have sections, like a box or cylinder, the function does not need to be updated.

B. Real-Time Geometry Manipulation

One feature of GLOVES that distinguishes it from other programs is its ability to directly manipulate the geometry in real-time. This is done through sensitivities. After a component is built or updated, the sensitivities of the component are computed with respect to its design parameters. In a UDP, a routine can be written to compute the sensitivities analytically. If no sensitivity routine is provided, they are computed via finite differences [5]. However, this process takes longer because a perturbation of the configuration must be generated for each of the design parameters. For now, there are no sensitivity routines associated with any of the UDPs that create GLOVES components. This is an area of future work that would allow GLOVES to operate more efficiently.

There are two steps to manipulate the geometry in real-time. The first step involves computing what the new value of the design parameter should be, and is completed in the browser. Equation 1 reveals how every point on the geometry is updated. In the equation, $\overline{x_0}$, $\dot{\overline{x}}$, and $\overline{x_{3D}}$ are the original position, sensitivity of the point with respect to the design parameter being modified, and the final position, respectively. The variable t represents a 'time' to scale the sensitivity by, and is found by performing a Golden Section Search on $d^2(t)$, the square of the distance between the 2D coordinates of the pip and 2D coordinates of the mouse, to minimize its value. The second step of manipulating the geometry involves re-drawing the component. This is done in the server to update the VBOs and in the browser to process the updated VBOs.

$$\overline{x_{3D}} = \overline{x_0} + \dot{\overline{x}}t \tag{1}$$

Before explaining the Golden Section Search algorithm, the steps to compute d^2 will be explained. The position of the mouse, $\overline{m}$, is recorded in 2D coordinates, or screen coordinates. Since the 3D coordinates of the pip are known, $\overline{p_{3D}}$, they can be transformed into 2D coordinates to compute the distance between itself and the mouse position. The first step of the transformation is to scale the 3D coordinate by the focus, a vector of 4 doubles, $\begin{bmatrix} c_x & c_y & c_z & s \end{bmatrix}$, and is shown in Equation 2. The first three entries of the vector correspond to the 3D coordinate at the center of the Graphics Window and the last entry corresponds to the 3D scale of the display.

$$\begin{aligned} p_{x,\text{scaled}} &= \frac{p_{x,3D} - c_x}{s} \\ p_{y,\text{scaled}} &= \frac{p_{y,3D} - c_y}{s} \\ p_{z,\text{scaled}} &= \frac{p_{z,3D} - c_z}{s} \end{aligned} \tag{2}$$

After the scaled coordinates of the pip are found, they are multiplied by a transformation matrix to convert them to the scaled 2D coordinates. The transformation matrix is normally used to convert from scaled 2D coordinates to scaled 3D coordinates, and is given in Equation 3. In the matrix, $\overline{R}$ is a 3-by-3 rotation matrix with orthonormal columns, and depends on how the user rotates the view. The diagonal of the rotation matrix is multiplied by a nonnegative scale factor, depending on how much the user has zoomed in/out. $\overline{\Delta}$ is a 3-by-1 column vector and depends on how much the user pans (translates) the view. $\overline{o}$ is a 3-by-1 column vector of offsets to illustrate 3D perspective on a 2D canvas. The last entry in the matrix, w, is a scalar value and represents an offset between the scaled 3D and scaled 2D widths of the display.

$$\overline{T} = \left[\begin{array}{c|c} \overline{R} & \overline{\Delta} \\ \hline \overline{o}^T & w \end{array}\right] \tag{3}$$

Since the described transformation matrix converts from scaled 2D coordinates to scaled 3D coordinates, $\overline{p_{\text{scaled}}}$, is multiplied by $\left(\overline{T}\right)^{-1}$ to obtain the scaled 2D coordinates of the pip. However, the transformation matrix is 4-by-4, but the

vector of scaled 3D coordinates is 3-by-1. To resolve this, the homogeneous coordinates of the scaled 3D coordinates, $\left[\overline{p_{\text{scaled}}} \quad 1\right]^T$, are multiplied by $\left(\overline{T}\right)^{-1}$.

$$\begin{bmatrix} p_{x,\text{2D}} \\ p_{y,\text{2D}} \\ p_{z,\text{2D}} \\ w_{\text{2D}} \end{bmatrix} = \left(\overline{T}\right)^{-1} \begin{bmatrix} p_{x,\text{scaled}} \\ p_{x,\text{scaled}} \\ p_{x,\text{scaled}} \\ 1 \end{bmatrix} \tag{4}$$

Now that the scaled 2D coordinates of the pip have been obtained, $p_{z,\text{2D}}$ can be disregarded because there is no 'depth' in the screen. To obtain the 2D coordinates of the pip, $p_{x,\text{2D}}$ and $p_{y,\text{2D}}$ are scaled by $(w_{\text{2D}})^{-1}$. Notice that this portion of the transformation is non-linear, which is one of the reasons why a Golden Section Search was chosen as the optimization technique.

$$\begin{aligned} p_{x,\text{2D}} &= \frac{p_{x,\text{scaled}}}{w_{\text{2D}}} \\ p_{y,\text{2D}} &= \frac{p_{y,\text{scaled}}}{w_{\text{2D}}} \end{aligned} \tag{5}$$

Lastly, d^2 can be computed as shown in Equation 6. Since the Golden Section Search is searching for the new value of the design parameter, it does not matter whether the distance between the pip and mouse is squared or not.

$$d^2 = \left(m_x - p_{x,\text{2D}}\right)^2 \left(m_y - p_{y,\text{2D}}\right)^2 \tag{6}$$

The Golden Section Search is performed as follows [8]:

1) Compute the initial position of the pip, $d^2(0)$.
2) Set $i = 0$.
3) Compute $\alpha_i = \delta \times 1.612^i$ and evaluate $d^2(-\alpha_i)$ and $d^2(+\alpha_i)$, where δ is a step size. In GLOVES, $\delta = 0.10$.
4) If the minimum is bounded, meaning that $d^2(0) < d^2(-\alpha_i)$ and $d^2(0) < d^2(+\alpha_i)$, continue. Otherwise, set $i = i + 1$ and return to Step 3.
5) Set $\alpha_l = -\alpha_i$, $\alpha_u = +\alpha_i$.
6) Compute the Interval of Uncertainty, $I = \alpha_u - \alpha_l$.
7) While $I > \epsilon$: (in GLOVES, $\epsilon = 10^{-6}$)
    a) Compute $\alpha_a = \alpha_l + 0.382I$ and $\alpha_b = \alpha_l + 0.618I$.
    b) Compute $d^2(\alpha_a)$ and $d^2(\alpha_b)$.
    c) If $d^2(\alpha_a) < d^2(\alpha_b)$, set $\alpha_u = \alpha_b$. Otherwise, set $\alpha_l = \alpha_a$.
    d) Compute the Interval of Uncertainty, $I = \alpha_u - \alpha_l$.
8) Set $t = 0.5(\alpha_u - \alpha_l)$ and end optimization.

Once the Golden Section Search has converged, the proposed design parameter value, v_{new} is computed from its original value, v_{old} and t found from minimizing the distance between the mouse and pip. The proposed design parameter value is displayed to the user on the 2D canvas rendering context.

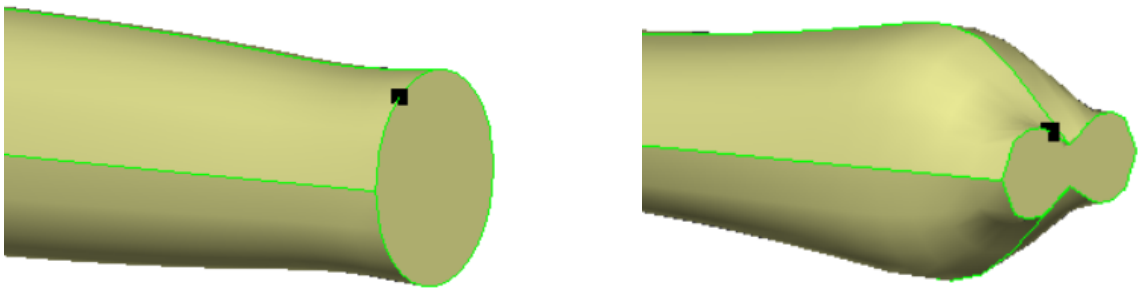
$$v_{\text{new}} = v_{\text{old}} + t \tag{7}$$

In order to update the geometry, the browser sends a message to the server containing the optimum t. The server then computes $\overline{x_{\text{3D}}}$ via Equation 1 for each point on the geometry, updates the respective VBOs, and then sends the VBOs to the browser to be re-drawn. Recall that the geometry is displayed on the 3D canvas rendering context.

Two examples of manipulating the geometry are shown in Figures 8 and 9, which show a box as its length changes and a fuselage as its y-radius at the tail-end changes, respectively. Notice that when manipulating the box, the modified geometry appears as the user would expect - the box has become longer and still looks like a box. However, when the fuselage is modified, the tail-end no longer appears as a super-ellipse, but rather as two teardrop shapes. This is because Equation 1 linearly approximates the new position of each point while the geometry is manipulated. The points on the box vary linearly with respect to its length. Thus, there is no discrepancy between the modified geometry and what it would look like after a re-build. However, the points on the fuselage vary non-linearly due to the blend. Hence, there is a discrepancy between the modified geometry and what it would look like after a re-build. If small (local) changes are made, the modified geometry should accurately represent what it would look like after a re-build. Otherwise, the geometry will likely appear distorted.

(a) Box before manipulation. **(b) Box during manipulation.**

Fig. 8 Modifying the length of the box.



(a) Fuselage end before manipulation. **(b) Fuselage end during manipulation.**

Fig. 9 Modifying the y-radius of a fuselage section.

C. Weight and Balance Analysis

Another feature that distinguishes GLOVES from other programs is its ability to perform a simple weight and balance analysis in real-time. Again, this can be done as a result of using sensitivities to modify the geometry. To accomplish this, the sensitivities of $x_{\mathrm{CG},i}$, $x_{\mathrm{CG},i}$, $x_{\mathrm{CG},i}$, the x-, y-, and z-centers of gravity of each component, respectively, and V_i, the volume of each component, are computed with respect to the modified design parameter. After this, the sensitivities of X_{CG}, Y_{CG}, and Z_{CG} the x-, y-, and z-centers of gravity of the layout, respectively, can be computed using Equation 8:

$$\dot{X}_{\text{CG}} = \frac{(\sum_{i=1}^{n} V_i)(\dot{x}_{\text{CG},j} V_j + x_{\text{CG},j} \dot{V}_j) - (\sum_{i=1}^{n} x_{\text{CG},i} V_i)(\dot{V}_j)}{(\sum_{i=1}^{n} V_i)^2}$$
$$\dot{Y}_{\text{CG}} = \frac{(\sum_{i=1}^{n} V_i)(\dot{y}_{\text{CG},j} V_j + x_{\text{CG},j} \dot{V}_j) - (\sum_{i=1}^{n} y_{\text{CG},i} V_i)(\dot{V}_j)}{(\sum_{i=1}^{n} V_i)^2} \tag{8}$$
$$\dot{Z}_{\text{CG}} = \frac{(\sum_{i=1}^{n} V_i)(\dot{z}_{\text{CG},j} V_j + z_{\text{CG},j} \dot{V}_j) - (\sum_{i=1}^{n} z_{\text{CG},i} V_i)(\dot{V}_j)}{(\sum_{i=1}^{n} V_i)^2}$$

where n represents the number of components in the layout and j represents the index of the component that is being manipulated.

A demonstration of a weight and balance analysis is shown. For this demonstration, there are two components: (1) a box and (2) a sphere. The initial configuration is depicted in Figure 10a. The CG of each component is plotted in red. The CG of the overall layout is plotted in blue. First, the depth of the box (in the z-direction) will be increased. As the depth of the box increases, the volume of the box increases, and the center of gravity of the layout shifts closer to the box, as seen in Figure 10b. However, notice that the center of gravity of the box does not change. This is because the box is being stretched equally in the $\pm z$-directions. Now, the sphere will be moved in the $+y$-direction by increasing its y-center. As the y-center of the sphere increases, its y-center of gravity increases, and the center of gravity of the also moves in the $+y$-direction, as seen in Figure 10c. This time, notice that the center of gravity of the sphere translated with the geometry.

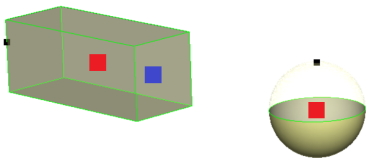
D. Component Linking

The final feature of GLOVES is its ability to link components together. The purpose of this feature is to connect one or more components that are dependent upon each other. This operation is done primarily in the browser. An example of when component linking would be useful is if a user was trying to find the minimum volume fuselage to enclose a payload. Another example would be connecting the horizontal tail to the main wing: as the area of the main wing changes, the horizontal tail volume coefficient needs to be changed. Component linking would also be useful if GLOVES were to contain a feature that allowed the automatic sizing of components. That way, the sizing of one component can depend on the size of other components in the layout.

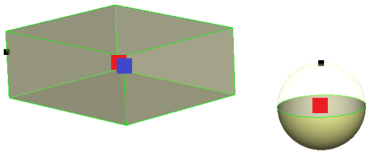
In order to connect multiple components, first select the 'parent' component (the component that will depend upon all other connections) by hovering over it with the mouse and pressing "c" on the keyboard. After pressing "c" the Key Window will change and the user can continue hovering over the 'children' components (the component that will impact the parent) and pressing "c" on the keyboard to connect them to the parent. Each time a connection is made, the user will see a message posted in the Message Window to confirm the selected children component. If a component is erroneously connected to another one, the user can press "r" to remove the connection. Similarly, the user will receive a message informing them of the connection that was removed.

After all necessary components are connected, the user will press "d" on the keyboard. At this time, a message will be sent to the server indicating which component is the parent and which ones are the children. The server will then add an attribute, `_.children._`, to all of the faces on the parent. The user can confirm that the connection was made by hovering or the parent component and pressing "6" or "^" on the keyboard. In the Message Window, the name of the component and its attributes will be listed.

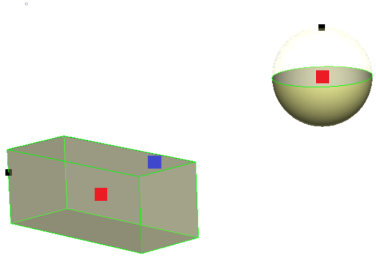
Figures 11a (before linking) and 11b (after linking) illustrate an example with three bodies: (1) a sphere, which will be the parent; (2) a box, which will be one of the children; and (3) a cylinder, which will also be one of the children. As seen in Figure 11b, there is a post in the Message Window with the attribute: `_.children._ myBox;myCyl`. This confirms that the connection was properly made between the parent and children.



(a) Layout and CGs before manipulation.

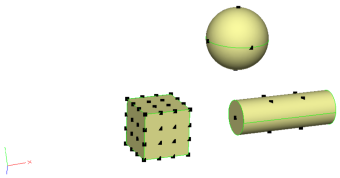


(b) Box stretched in the $\pm z$-direction.



(c) Sphere moved in the $+y$-direction.

Fig. 10 Weight and balance analysis of a simple layout.



(a) GLOVES before component linking.

```
Picked: mySphr Face 2
        _.pips._=  0.500000 0.500000
        _.type._=  NULL
        _body=  21 2
         brch=  29 2
        _name=  mySphr
        _.children._=  myBox;myCyl
        _hist=  21
```

(b) Message window after components were linked.

Fig. 11 Component linking between a sphere (parent), box (child), and cylinder (child).

VI. Demonstration

To demonstrate the capabilities of the new version of GLOVES, a simple transport layout is made. Select screenshots of the demonstration are presented in the figures below. First, launch ESP with a blank CSM file. If the file has not already been saved, do so before launching GLOVES. Then, the user left-clicks the "Tool" option in the tree window and then clicks "Gloves" (shown previously in Figure 5). This launches GLOVES, and the user can now begin to add components to the layout. If the reader wants to replicate this configuration, the components and their respective design parameters are listed in the Appendix.

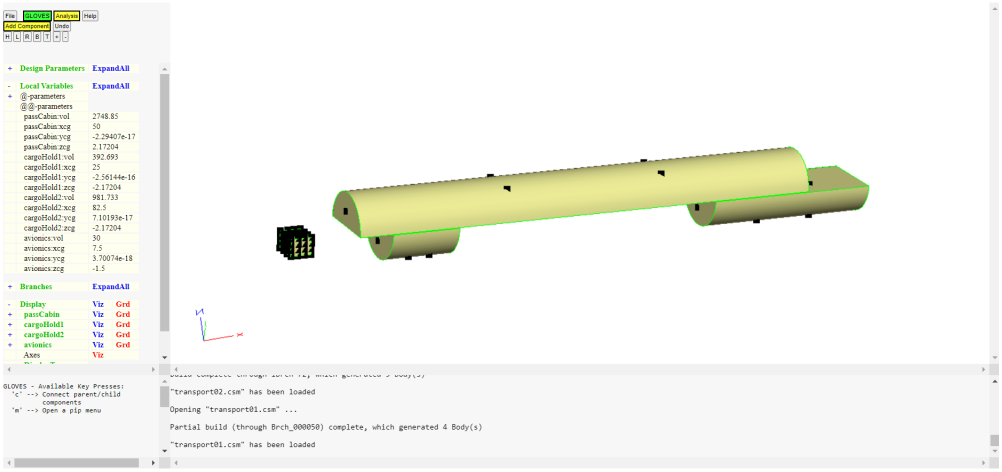


Fig. 12 Payload arranged in GLOVES.

First, the payload is arranged on the screen. This is done with three cylinders and a box. The cylinders represent: (1) the passenger cabin, the half cylinder on top of the two smaller cylinder halves; (2) the front cargo hold, the half cylinder on the bottom left of the configuration; and (3) the aft cargo hold, the half cylinder on the bottom right of the configuration. The box represents the avionics, and is forward of the passenger cabin and cargo holds. The resulting payload arrangement is depicted in Figure 12. After this phase, the user navigates to and left-clicks the "Gloves" button and then left-clicks "Save". After saving, the configuration is appended to the current CSM file. The user must edit the CSM file and move the statements associated with GLOVES before the end of the file. This can be done by left-clicking "File", left-clicking "Edit", and then moving the necessary code. Note that there is a header, `# begin GLOVES`, and a footer, `# end GLOVES`, that must remain in the CSM file. This is how GLOVES knows which components should be loaded into the next session of GLOVES.

Second, the fuselage is created to fully enclose the payload. Before adding another component, the user initializes another session of GLOVES. This is done by left-clicking on "Tool" and then left-clicking "Gloves". If the header, `# begin GLOVES`, and the footer, `# end GLOVES`, are in the CSM script, GLOVES will automatically load the components between the header and footer. After loading the components and computing the sensitivities of each component with respect to its design parameters, the user can add a fuselage component. The fuselage is pointed at the nose and blunt at its tail end. That way, in a more detailed model of the transport, an Auxiliary Power Unit can be placed in the aft portion of the fuselage. The resulting fuselage is depicted in Figure 13. Some of the faces of the fuselage were set to be transparent and the pips were turned off so the reader could see the payload inside of it.

After the fuselage is added, the user saves the configuration. Again, this is done by left-clicking "Gloves", then "Save". If a header for GLOVES is found in the CSM file, all of the existing components in the CSM file are overwritten and the new components are written to the CSM file. As a caution to the reader, any comments added in-between the GLOVES header and footer are erased from the CSM file upon saving and exiting GLOVES. If the user quits from GLOVES without saving, the CSM file is not altered.

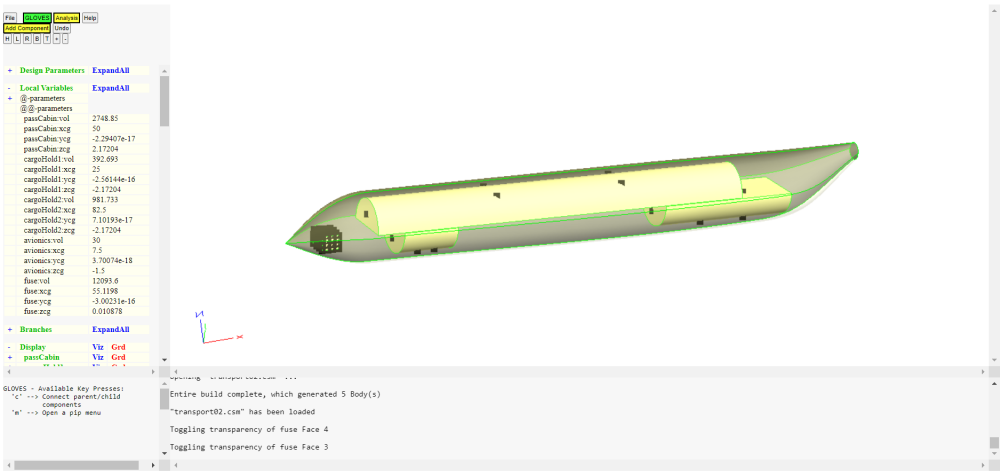


Fig. 13 Fuselage made to surround payload.

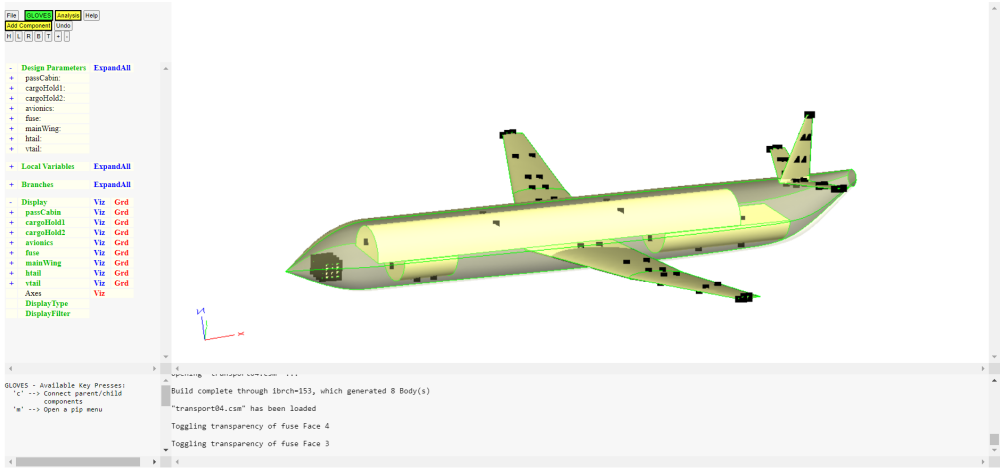


Fig. 14 Wing, horizontal tail, and vertical tail added to the configuration.

Next, the wing, horizontal tail, and vertical tail components are added to the configuration. Note that the wing and horizontal tail components are the same. Since the predefined wing component is made of three sections, the middle section is removed such that the horizontal tail is made of only two airfoils. The final configuration is depicted in Figure 14. Again, note that some of the faces of the fuselage were set to be transparent and the pips were turned off so the reader could see the payload inside of it.

VII. Summary

In this research, a new version of GLOVES for ESP was developed. This enhanced computational environment allows a user to interactively arrange and size aircraft components and directly manipulate their geometry. In order to manipulate the geometry without constantly re-building it, ESP's sensitivity capabilities were used. These capabilities were also used to include a weight and balance analysis tool, which illustrates how the component (local) and layout (global) CGs are affected by changing the value of a design parameter. To demonstrate the current capabilities of GLOVES, a transport layout was created and a weight and balance analysis was performed whilst arranging the payload.

It is much simpler to manipulate the geometry directly than it is to use sliders. By doing so, the user only needs to look at one window to understand how modifying the design parameter affects the geometry. Furthermore, the

user is aware of the design parameters that affect the component the most because they are listed in a menu after a pip is selected. An advantage of using sensitivities to do this is that the user receives immediate feedback on how their proposed design change affects the layout. However, if the changes made to the configuration are 'too large', the modified geometry may not accurately represent what the geometry will look like after it is re-built. If the user makes small, 'local' changes to the geometry, the feedback they receive is accurate.

Aside from the user experience manipulating the geometry, one downfall of using sensitivities is that the times to re-build the configuration can be lengthy. For now, it was decided that all of the sensitivities would be computed via finite differences rather than analytically. That way, attention could be focused on building up the capabilities of GLOVES rather than writing sensitivity routines and improving its efficiency. As a result of this, a perturbed model must be built for each design parameter in the configuration so the sensitivities to be computed. In the future, once sensitivities are computed analytically, the process will be faster. For the transport configuration exhibited in the demonstration, the time to build the configuration when launching ESP was 2.90 seconds on a Windows 10 Computer with an Intel i7 Processor. However, when loading the configuration into GLOVES, it took 214 seconds. During this time, the model was built and sensitivities were computed with respect to 121 design parameters. For comparison, modifying the fuselage, which had the most design parameters (39), took 35.9 seconds to build the perturbed models and compute the sensitivities.

VIII. Future Work

There are multiple areas of improvement for future iterations of GLOVES. The first area of improvement is to write sensitivity routines for all of the existing GLOVES components. That way, the time to compute the sensitivities with respect to each design parameter is reduced. This will also allow the user to spend more time interacting with the geometry and improving the aircraft layout. The second area of improvement is to allow the user to make multiple changes to a component before it is re-built. This reduces the number of times that the sensitivities need to be computed and allows the user to work more efficiently while developing the layout. For example, if a user is going to change the y- and z-radii of a fuselage, it makes more sense to compute the sensitivities and update the model after both changes have been made. Another example involves the user translating a component in multiple directions. Translating a component will not affect the sensitivities of the component. However, GLOVES requires that the sensitivities be re-computed because the value of a design parameter changed. To remedy this, it may be useful to include an "Update Component" button, which allows the user to request that the sensitivities of one or more components be re-computed and that the menus on the pips be updated.

The third area of improvement is to include additional systems that the user can add components from: (1) the Propulsion System; (2) the Control System; and (3) the Structural System. The Propulsion System would contain turboshaft, turboprop, and turbojet engines, nacelles, pods/pylons to connect the propulsion system to the main wing or fuselage. The Control System would contain flaps/slats, ailerons/elevators, and rudders. The Structural System would contain ribs, spars, bulkheads, and longerons. The final area of improvement is to implement a feature that allows a user to automatically size different components of the layout. Two examples of this include: (1) finding a minimum volume fuselage to enclose a payload arrangement; or (2) computing the area of a wing based on the weight of the payload, desired lift coefficient, and expected cruise velocity. That way, the user can focus on the parent-child relationships of components and design parameters across multiple, high-level aerospace systems.

Appendix

The following is a list of all the components needed to re-create the transport example in Section VI. The values of the design and configuration parameters for each component are listed under its name as CSM statements. However, these statements should not be used in an attempt to load the configuration into GLOVES. Instead, it should be built on its own using GLOVES.

- Passenger Cabin:

```
DESPMTR  xbeg       15.00
DESPMTR  ybeg        0.00
DESPMTR  zbeg        0.05
DESPMTR  length     70.00
DESPMTR  azimuth     0.00
DESPMTR  elevation   0.00
DESPMTR  rad         5.00
CFGPMTR  half           2
```

- Forward Cargo Hold:

```
DESPMTR  xbeg       20.00
DESPMTR  ybeg        0.00
DESPMTR  zbeg       -0.05
DESPMTR  length     10.00
DESPMTR  azimuth     0.00
DESPMTR  elevation   0.00
DESPMTR  rad         5.00
CFGPMTR  half           1
```

- Aft Cargo Hold:

```
DESPMTR  xbeg       70.00
DESPMTR  ybeg        0.00
DESPMTR  zbeg       -0.05
DESPMTR  length     25.00
DESPMTR  azimuth     0.00
DESPMTR  elevation   0.00
DESPMTR  rad         5.00
CFGPMTR  half           1
```

- Avionics:

```
DESPMTR  xcent  15.00
DESPMTR  ycent   0.00
DESPMTR  zcent   0.05
DESPMTR  dx     70.00
DESPMTR  dy      0.00
DESPMTR  dz      0.00
```

- Fuselage:

```
DESPMTR  length                          100.00
DESPMTR  width                            12.50
DESPMTR  height                           12.50
DESPMTR  xcent     0.00;0.05;0.15;0.90;0.98;1.10
DESPMTR  ycent     0.00;0.00;0.00;0.00;0.00;0.00
DESPMTR  zcent   -1.50;-0.60;0.00;0.00;0.60;5.00
DESPMTR  ry        0.00;0.50;1.00;1.00;0.90;0.25
DESPMTR  rz        0.00;0.50;1.00;1.00;0.90;0.25
DESPMTR  power     2.00;2.00;2.00;2.00;2.00;2.00
CFGPMTR  curv                      2;2;1;2;2;2
```

- Wing:

```
DESPMTR  area           1000.00
DESPMTR  aspect            9.00
DESPMTR  xroot            45.00
DESPMTR  yroot             0.00
DESPMTR  zroot            -2.00
DESPMTR  taper         0.60;0.40
DESPMTR  sweep      20.00;20.00
DESPMTR  dihed         4.00;8.00
DESPMTR  yloc    0.00;0.50;1.00
DESPMTR  camber  0.02;0.02;0.02
DESPMTR  maxloc  0.40;0.40;0.40
DESPMTR  thick   0.12;0.12;0.12
DESPMTR  alpha   0.00;0.00;0.00
CFGPMTR  sharp                1
CFGPMTR  curv              0;0;2
```

- Horizontal Tail:

```
DESPMTR  area       100.00
DESPMTR  aspect       6.00
DESPMTR  xroot       95.00
DESPMTR  yroot        0.00
DESPMTR  zroot        5.50
DESPMTR  taper        0.60
DESPMTR  sweep       20.00
DESPMTR  dihed       10.00
DESPMTR  yloc    0.00;1.00
DESPMTR  camber  0.00;0.00
DESPMTR  maxloc  0.40;0.40
DESPMTR  thick   0.12;0.12
DESPMTR  alpha   0.00;0.00
CFGPMTR  sharp          1
CFGPMTR  curv         0;2
```

- Vertical Tail:

```
DESPMTR  area        50.00
DESPMTR  aspect       3.00
DESPMTR  xroot       95.00
DESPMTR  yroot        0.00
DESPMTR  zroot        5.50
DESPMTR  taper        0.20
DESPMTR  sweep       35.00
DESPMTR  dihed        0.00
DESPMTR  zloc    0.00;1.00
DESPMTR  camber  0.00;0.00
DESPMTR  maxloc  0.00;0.00
DESPMTR  thick   0.12;0.12
DESPMTR  alpha   0.00;0.00
CFGPMTR  sharp          1
CFGPMTR  curv         2;2
```


Acknowledgments

This work was partially funded by the EnCAPS Project, AFRL Contract FA8650-20-2-2002: "Enhanced CAPS"; Ryan Durscher was the technical monitor.

References

[1] Litherland, B., "OpenVSP Ground School," , 2021. URL https://vspu.larc.nasa.gov/.

[2] NASA, "Open Vehicle Sketch Pad," , 2020. URL https://software.nasa.gov/software/LAR-17491-1, version 3.23.0.

[3] McDonald, R. A., and R., G. J., "Open Vehicle Sketch Pad: An Open Source Parametric Geometry and Analysis Tool for Conceptual Aircraft Design," *AIAA 2022-0004*, 2022.

[4] Dannenhoffer, J., and Haimes, R., "The Engineering Sketch Pad," , 2021. URL https://acdl.mit.edu/esp/, version 1.20.

[5] Dannenhoffer, J., and Haimes, R., "Engineering Sketch Pad Training Session 1," PDF, 2021.

[6] Dannenhoffer, J., and Haimes, R., "Engineering Sketch Pad Training Session 4," PDF, 2021.

[7] Dannenhoffer, J., and Haimes, R., "Engineering Sketch Pad Training Session 3," PDF, 2021.

[8] Arora, J. S., *Introduction to Optimum Design*, 4th ed., Academic Press, 2017.