

ABSTRACT

Allen, Jeremiah Daniel. M.S.M.E, Department of Mechanical and Materials Engineering, Wright State University, 2005. Air Vehicle Building.

Developing a new air vehicle involves a large amount of design concepts and stages. Quite often, many design tools are used to complete a design. One tool is used to develop the parametric geometry model, while another tool is used to complete aerodynamic analysis, and another structural analysis. Learning, using, and integrating these utilities proves to be a difficult task. First a CAD program is used to generate a parametric model of an air vehicle. Once completed, the model is processed in an aerodynamic code and analysis is run. Loads produced from this analysis are translated into structural loads, and finite element analysis is done. In order to complete the design, the user has to learn how to apply and integrate each program which is a demanding task. In this research an application called Air Vehicle Builder (AVB) has been researched and proven to be a great collaborative tool. A novice or senior designer will find the AVB has a small learning curve and is useful in many areas of aircraft design. It allows quick baseline modeling of an aircraft, and has the ability to be used to complete an optimized aircraft design. Furthermore, the AVB can create a mesh for whatever structure imagined, and output data for design modules such as NASTRAN, so that the user does not have to learn integration. Plus, if a procedure or formula desired is not available within the AVB, the AVB integrator is set up to allow a user to incorporate his/her own code in order to use the desired formula. This integration process is made easy by the AVB.

Table of Contents

1. INTRODUCTION.....	1
2. THE MODELING LANGUAGE	11
2.1 Adaptive Modeling Language.....	11
2.2 The Air Vehicle Builder.....	12
2.2.1 Air Vehicle Builder Environment.....	13
2.2.1.1 Model Tree.....	14
2.2.1.2 Inspect Object	15
2.2.1.3 Work Area.....	16
2.2.2 Using the AVB.....	17
2.2.3 Building a Wing.....	19
2.2.4 Building a Fuselage.....	24
2.2.4.1 Drawing the Fuselage	26
2.2.5 Other Tools	27
2.2.5.1 Connecting Structures to the Aircraft	27
3. INTEGRATION	28
3.1 Integration Background	28
3.2 External Programs.....	28
3.2.1 Integration Example.....	28
3.2.1.1 Integration Scenario	29
3.2.1.2 Integration Environment	29
3.2.1.2.1 Properties-file-io.aml File	30
3.2.1.2.2 Properties List	31
3.2.1.2.3 Writing to a Text File.....	32
3.2.1.2.4 Reading a Text File.....	32
3.2.1.2.5 Batch File	32
3.2.1.2.6 Emacs Command Window	33
3.2.1.2.7 Integration Procedure.....	34
3.2.1.2.8 Integration Run	35
3.2.1.2.9 Integration Usefulness	35
3.3 CAD Programs.....	36
3.3.1 Exporting Example	36
3.3.1.1 Exporting A Model	36
3.3.3 CAD Program Usefulness.....	37
4. FINITE ELEMENT ANALYSIS	38
4.1 Finite Element Analysis.....	38
4.1.1 Finite Element Tools.....	38
4.1.2 Finite Element Options	39
4.1.3 Aerodynamic Analysis.....	40
4.1.3.1 Aerodynamic Panel Model	40
4.1.3.2 Aerodynamic Properties.....	40
4.1.4 Meshing.....	41
4.1.5 Finite Element and Aerodynamic Integration.....	45
4.1.6 Reading and Post-processing Files	46
5. OPTIMIZATION.....	48

5.1 Optimization Introduction.....	48
5.2 NASTRAN File	48
5.3 Optimization Environment.....	49
5.3.1 AMOPT Toolbox.....	49
5.3.1.1 AMOPT Toolbar.....	49
5.3.1.2 Optimization Object.....	50
5.3.1.3 Optimization Run.....	56
5.4 Optimization Studies.....	57
5.4.1 Sensitivity Analysis	57
5.4.1.1 Setting Up Sensitivity Analysis	57
5.4.1.2 Sensitivity Study Problem Statement.....	58
5.4.1.3 Sensitivity Study Wing Model.....	58
5.4.1.4 Sensitivity Problem Setup.....	59
5.4.1.5 Sweep Angle Sensitivity Study.....	59
5.4.1.6 Chord Length Sensitivity Study.....	61
5.4.1.7 Span Sensitivity Study	63
5.4.1.8 Sensitivity Study Conclusion.....	64
5.4.2 DOE Analysis	65
5.4.2.1 Setting Up DOE	65
5.4.2.2 DOE Problem Statement.....	66
5.4.2.3 DOE Wing Model.....	66
5.4.2.4 DOE Wing Design Variables.....	67
5.4.2.5 DOE Wing Design Constraints.....	67
5.4.2.6 DOE Wing Objective.....	68
5.4.2.7 DOE Problem Setup.....	68
5.4.2.8 DOE Run Matrix.....	68
5.4.2.9 DOE Results.....	70
5.4.2.9 Regression Analysis of DOE Results	70
5.4.2.10 Using The Solver	71
5.4.2.11 DOE Conclusion	72
5.4.3 DOT Optimization	73
5.4.3.1 Setting Up DOT Optimization.....	73
5.4.3.2 DOT Optimization Problem Statement.....	74
5.4.3.3 DOT Optimization Wing Model.....	74
5.4.3.4 DOT Optimization Problem Set Up.....	75
5.4.3.5 DOT Optimization Design Variables.....	75
5.4.3.6 DOT Optimization Design Constraints.....	75
5.4.3.7 DOT Optimization Results	76
5.4.3.8 DOT Optimization Conclusion.....	78
5.5 Model Validation	78
Figure 5-19: Comparison Model.....	79
5.5.1 Configuration Change Check.....	79
5.6 Optimization Conclusion	80
6. SUMMARY	81
6.1 Summary.....	81
7. FUTURE WORK.....	82

7.1 Future Work	82
8. BIBLIOGRAPHY.....	83
Appendix A	
Appendix B	

List of Figures

Figure 1-1: AVB Rib and Spar Model.....	9
Figure 2-1: AVB GUI.....	13
Figure 2-2: Model Tree.....	14
Figure 2-3: Inspect Object Menu.....	16
Figure 2-4: Work Area.....	17
Figure 2-5: AMRaven Tool Bar.....	17
Figure 2-6: Air Vehicle Hierarchy.....	18
Figure 2-7: Configuration Tab.....	20
Figure 2-8: Layout Tab.....	21
Figure 2-9: Substructures Tab.....	22
Figure 2-10: CS Substructures Tab.....	23
Figure 2-11: Attachments Tab.....	24
Figure 2-12: Fuselage Configuration Tab.....	25
Figure 2-13: Fuselage Substructures Tab.....	25
Figure 2-14: Primitive Menu Options.....	26
Figure 3-1: Wing Model Tree.....	30
Figure 3-2: Emacs Window.....	34
Figure 3-3: IDEAS Wing Model.....	36
Figure 3-4: AML Menu Toolbar.....	37
Table 4-1: AMRaven NASTRAN Cards.....	39
Figure 4-1: Analysis Types.....	40
Figure 4-2: Aerodynamic Properties.....	41
Figure 4-3: Meshing Path.....	42
Figure 4-4: Wing Mesh.....	43
Figure 4-5: Wing Mesh Case One.....	44
Figure 4-6: Wing Mesh Case Two.....	44
Figure 4-7: Deformation Plot.....	46
Figure 5-1: NASTRAN File Creation.....	48
Figure 5-3: AMOPT Toolbar.....	49
Figure 5-4: Exploration Toolbox.....	50
Figure 5-5: Design Variable Window.....	51
Figure 5-6: Constraint Window.....	52
Figure 5-7: Objective Window.....	53
Figure 5-8: Properties Tab.....	54
Figure 5-9: Run Tab.....	55
Figure 5-10: Plots Tab.....	56
Figure 5-11: BWB Wing Planform and Section Info.....	59
Table 5-1: Sweep Angle Constraints.....	60
Figure 5-12: Sweep Angle Sensitivity Table.....	60
Table 5-2: Sweep Angle Sensitivity Study Results.....	61
Table 5-3: Chord Length Study Constraints.....	61
Figure 5-13: Optimization Setup.....	62
Table 5-4: Chord Length Sensitivity Study Results.....	63
Table 5-5: Span Constraints.....	64

Table 5-6: Span Sensitivity Study Results.....	64
Figure 5-14: Trapezoid Wing Planform.....	66
Table 5-7: Trapezoidal Wing Dimensions.....	67
Table 5-8: DOE Design Constraints	67
Figure 5-15: DOE Run Matrix.....	69
Table 5-9: DOE Results.....	70
Table 5-10: Regression Analysis Coefficients.....	71
Table 5-11: DOE Solver Results.....	71
Figure 5-16: DOE Optimized Model	72
Figure 5-17: DOT Properties Tab.....	74
Table 5-12: DOT Optimization Design Variables	75
Table 5-13: DOT Optimization Design Constraints	75
Table 5-14: DOT Optimization Run Matrix	77
Figure 5-18: Optimized Trapezoid Wing Planform.....	78
Figure 5-19: Comparison Model.....	79
Figure 5-20: Altered Model	80

ACKNOWLEDGEMENT

The author wishes to express his gratitude and appreciation to Dr. Ramana V. Grandhi for constant guidance throughout the graduate studies. He would also like to thank friends and family for their continuous moral support throughout the course of the research effort.

This research is based upon work supported, in part, by the Wright Brother Institution. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the sponsors.

1. INTRODUCTION

Developing a new air vehicle involves a large amount of design concepts and stages. Quite often, many design tools are used to complete a design. One tool is used to develop the parametric geometry model, while another tool is used to complete aerodynamic analysis, and another structural analysis. Learning, using, and integrating these utilities proves to be a difficult task. First a CAD program is used to generate a parametric model of an air vehicle. Once completed, the model is processed in an aerodynamic code and analysis is run. Loads produced from this analysis are translated into structural loads, and finite element analysis is done. In order to complete the design, the user has to learn how to apply and integrate each program, which is a demanding task.

Current Design Tools

Analysis	Tools
Parametric Modeling	Unigraphics, PRO-E
Aerodynamic Analysis	PANAIR, ZAERO
Structural Analysis	NASTRAN, I-DEAS
Optimization	NASTRAN, Minitab

The United States Air Force has set up a Simulation-Based Research and Development (SBRD) team in an effort that seeks to improve the assessment method for new technologies by enabling improved connectivity between people and analysis tools. It seeks to implement commercial-off-the-shelf software whenever possible, with hopes of integrating various programs. The SBRD team should create an application that has a small learning curve so that users of all backgrounds will find it useful.

With these goals in mind this work unifies the air vehicle design process through integration of vehicle analysis codes into a single environment. The objective is to use the AVB for practical problems that are encountered in aircraft design and evaluate if it is successful in meeting the USAF SBRD goals. The goals investigated are as follows:

1. Is the AVB simple to learn and use? Is the AVB environment advantageous for quick learning and utilization?
2. Is the AVB environment set up to integrate with other programs? Can a model created in the AVB incorporate other programs if-needed to complete an aircraft design?
3. Does the AVB include finite element analysis capability? Is the finite element ability applicable to aircraft design? Is the finite element analysis tool compatible with current design modules like NASTRAN and PATRAN?
4. Can the AVB be used with optimization algorithms in order to fully optimize a design? Does the AVB contain ability to complete sensitivity analysis, design of experiments (DOE), and use optimization algorithms like feasible directions?

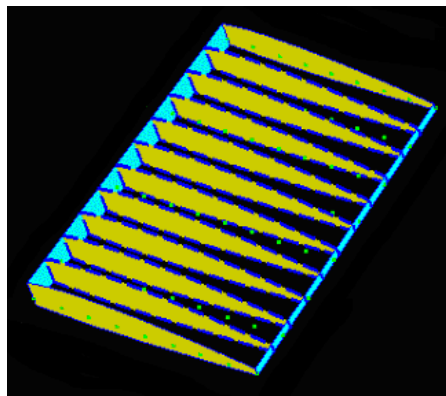


Figure 1-1: AVB Rib and Spar Model

This thesis answers all the above questions by using the AVB in real aircraft design problems. It starts with a brief discussion of current design methodologies and proceeds to illustrate the usefulness of the AVB in aircraft modeling, integration, finite element analysis, and optimization. It includes examples of all the above mentioned tools and incorporates the feasible direction optimization method to optimize an aircraft wing. It also expresses future work that could be completed to further investigate and improve the AVB, along with containing a User Manual illustrating how to use the AVB. The final chapter will summarize how the AVB has a small learning curve so that beginning aircraft designers will find it easy to use. It allows quick baseline modeling of an airplane, and can be employed to tie analysis codes together to complete a design of an aircraft, including optimization.

2. THE MODELING LANGUAGE

2.1 Adaptive Modeling Language

For this effort the Adaptive Modeling Language (AML) has been used to create an air vehicle modeling and design platform. AML is a framework for Knowledge Based Engineering that provides the ability of data transfer between codes. AML is built using the Allegro Common LISP programming language. Its object-oriented structure, hierarchical tree in model building, unified part model, dependency tracking, demand driven, and geometric capabilities make it a great language to set up a Graphical User Interface (GUI) for air vehicle building.

The current version of AML has a variety of features that make it useful in air vehicle design. First, AML is based on the concept of object-oriented programming. In object-oriented programming, the building blocks of applications are objects; they are not procedures or functions [5]. A consensus has been reached in the software industry that object-oriented programming is vital for ease of software development and reuse [4]. Object-oriented programming helps the programmer to organize the information that is required to do an analysis or design. For example, an object such as an airplane would include subobjects such as wing, tail, and fuselage. The wing object in turn could contain subobjects such as ribs, spars, control surfaces, and properties such as span and chord length [2]. This helps organize the properties of the air vehicle, and proves the collaborative nature of AML. For instance, even though airplanes like the F-16 or B-2 bomber may have different number of ribs and spars, they both have ribs and spars, so the object containing ribs and spars could be used for both air vehicles.

A second important feature of AML is its hierarchical tree in model building. Due to the fact that AML is an object-oriented language it contains a parent/child relationship amongst its objects. As stated before, the parent object would be the airplane, and the children of the airplane could be sub-objects like the wing or engine. The sub-object wing could have children sub-objects such as ribs and spars. What is nice about this feature is that it allows the user to add, edit, or delete objects and their properties regardless of what the order of their instantiation.

This hierarchical tree structure also contains a Unified Part Model paradigm. This structure allows the model of a given component, the wing for example, to contain all the data about the wing that will be required by the various analyses. For instance, the wing model could contain a panel aerodynamic model needed for low-speed calculation, and contain a finite element model of the wing box used for structural analysis [4]. This paradigm allows the model to grow as the design matures and new parts are created or new analyses are required. This simplifies the “bookkeeping” of the data, as AML has a built-in dependency tracking and demand driven calculation capabilities. The dependency tracking capability keeps track of every object that is dependent upon another object. Thus, if a rib is added to the rib object, the object that calculates weight will be changed as well. Furthermore, if the user wanted to calculate the weight, he or she can “demand” the value and it will know to retrieve the number of ribs and their mass properties to calculate the weight.

AML also has great geometry modeling capabilities. It contains geometric objects such as points, lines, curves, sheets, and boxes. It also has the capability to complete complex Boolean operations, do interpolated curves, and make Non-Uniform Ration B-spline (NURB) surfaces. NURB surfaces are a standard way of fitting 3-dimensional surfaces to a collection of points. These geometric capabilities allow for the calculation of properties such as volume, surface area, curvature, and many more.

All of the above mention capabilities combine to make AML an exceptional programming language to build a Graphical User Interface (GUI) to design air vehicles. The Air Vehicle Builder (AVB) that is used in this research study consists of such AML objects.

2.2 The Air Vehicle Builder

AML consist of objects within it that can be used to create a GUI for aircraft design. The AVB created by Technosoft, Inc., using AML, is such a GUI. It is an easy to use program that allows an aircraft designer to model an airplane without having to understand complicated programming within AML. Whether it is a senior or beginning level aircraft engineer, the AVB is a tool that all users will find simple to use, and

practical in all areas of aircraft design. It allows quicker baseline modeling of aircraft compared to many programs, including AML by itself. It has the ability to be used to complete an optimized aircraft design. Furthermore, the AVB can create a mesh for whatever structure imagined, and output data for design modules such as ASTROS or NASTRAN. Plus, if a procedure or formula desired is not available within the AVB, the AVB integrator is set up to allow a user to incorporate his/her own code in order to use the desired formula. A strong advantage of the AVB is that it integrates all the design procedures needed for aircraft design into one environment. It prevents a designer from having to learn using and integrating current tools, which can prove to be a difficult and timely task.

2.2.1 Air Vehicle Builder Environment

The AVB environment is very user friendly and helpful in aircraft design. Figure 2-1 illustrates the AVB GUI and its graphics.

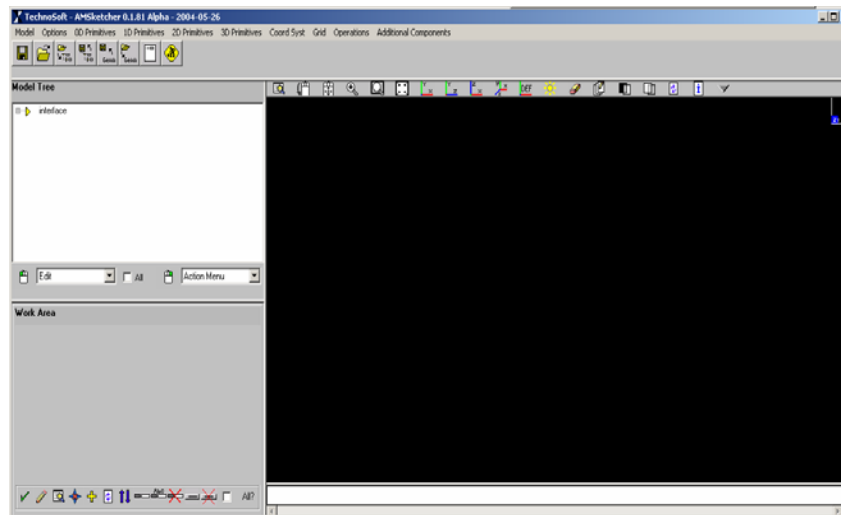


Figure 2-1: AVB GUI

Just like any other Computer Aided Drafting (CAD) system, the AVB has the general push-button options such as zoom, delete, regenerate, as well as different

coordinate system views. One of the helpful features in the AVB is the explanation blocks that display the button utility when the user holds the cursor over the button.

2.2.1.1 Model Tree

Another positive function in the AVB is the Model Tree. The Model Tree within the AVB keeps track of the design history of the model. As explained before, AML is an object-oriented language containing parent/child relationships. The Model Tree organizes these relationships by placing the parent object at the top, and the child objects underneath it. Figure 2-2 expresses this hierarchical relationship. Notice that there is a parent object called “air-vehicle-0001” and child objects called “design and analysis.” This hierarchical tree keeps the design history very organized so that the user can find a needed object quickly.

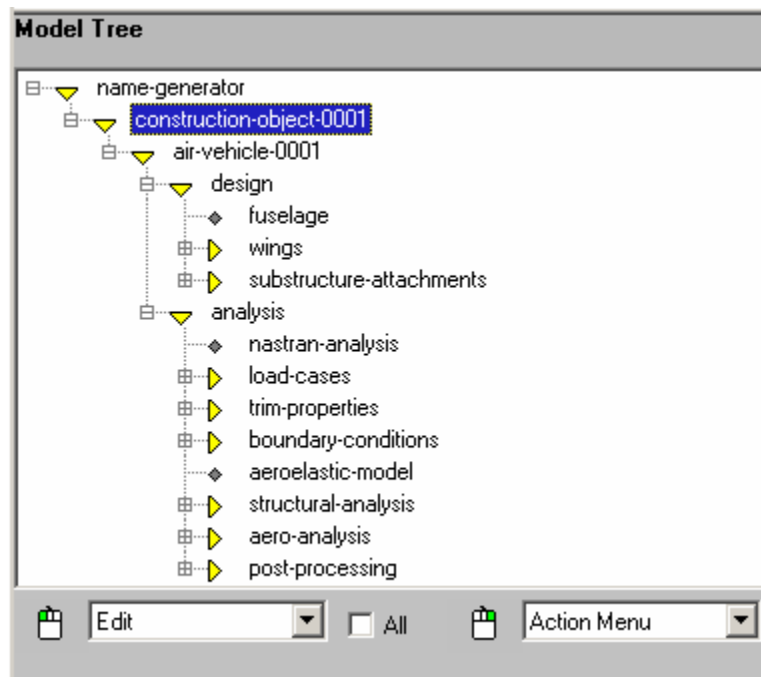


Figure 2-2: Model Tree

A nice option in the Model Tree is the ability to select an object within it and modify its properties. The user can edit or inspect the object individually without having to edit AML code. In other words, the AVB allows the user to select an object within the Model Tree, change some of its property values, update the object to the new values, and redraw the model on the screen with the new values. This is done without having to change the code within AML. This is a savings in the time and energy needed to understand the AML programming language. Other design programs like an Excel macro would require the programmer to change the code and then rerun.

2.2.1.2 Inspect Object

One of the options within the Model Tree is to inspect an object. This is a useful function because it enables the user to have access to all of the properties of the object and to demand their values. For example, if the user desired to see how many wing sections there are on the wing, the user could inspect the wing object and find the wing sections variable. Furthermore, the Inspect Object menu allows the designer to change property values within the object and to update the model. Figure 2-3 shows an example of the Inspect Object menu.

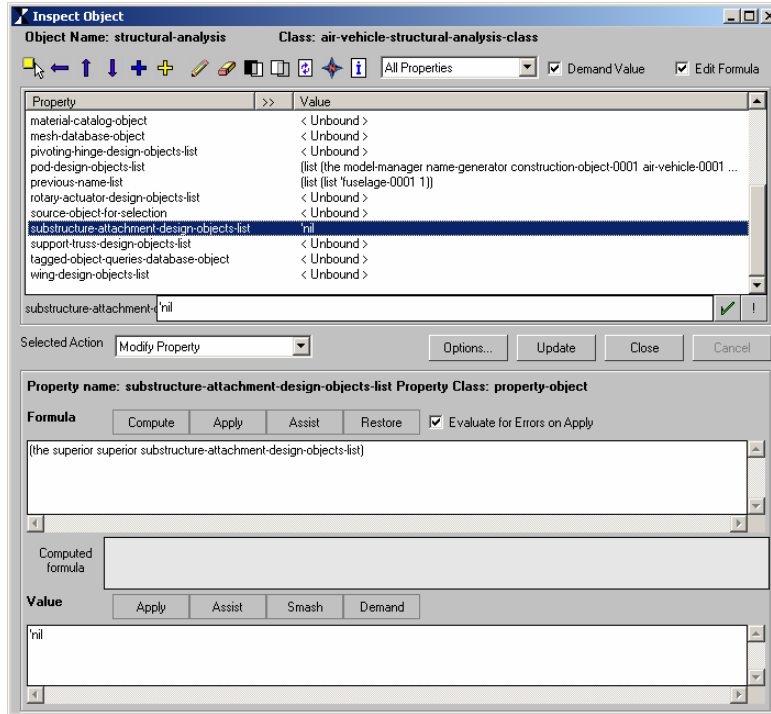


Figure 2-3: Inspect Object Menu

2.2.1.3 Work Area

When using the Model Tree to modify an object, some objects have property values that show up in the Work Area, which allows faster modification of the variables. For example, if a user is changing a coordinate system so that its origin changes, then the origin variable is available in the work area. Thus, instead of having to inspect the object and changing values within the Inspect Object menu, the user can adjust the values in the Work Area. Therefore, this function saves time and energy, and makes the system simpler to use. Furthermore, the AVB places the most often changed properties in the Work Area so that the variables are readily available to the user. In other words, one of the most common properties changed on a coordinate system is its origin. Thus, the AVB would have that variable obtainable in the Work Area. Figure 2-4 illustrates the Work Area.

constructs the tree in a logical order for design purposes. The idea behind this is to make modeling simpler and quicker. For example, in other CAD programs the user would have to model each aircraft component individually from scratch. However, with the AVB structure, the needed objects are already placed in the model, and all that the user needs to do is change the properties to the desired values. Figure 2-6 illustrates some of the tree structure in the Model Tree by using the Air Vehicle, Fuselage, and Wing buttons on the AMRaven menu.

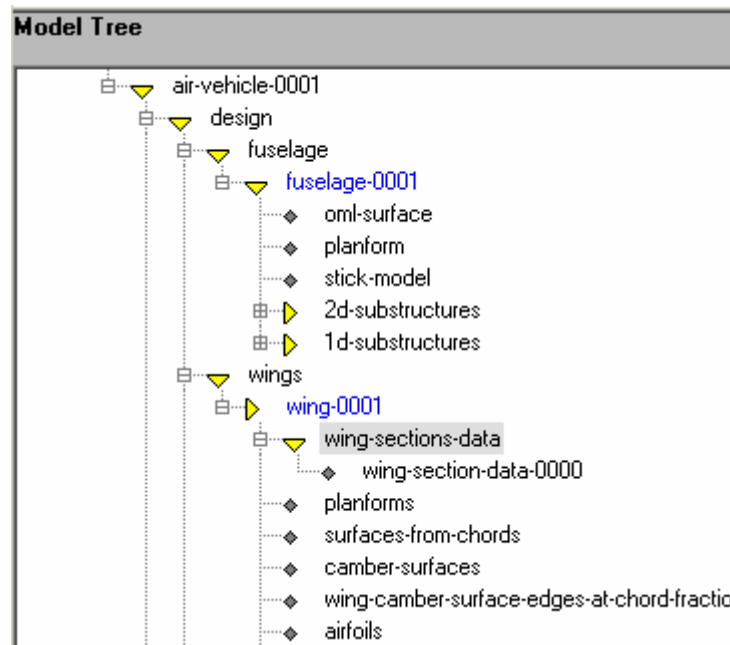


Figure 2-6: Air Vehicle Hierarchy

Notice how that the “air-vehicle-0001” object there is a child called “design” (there is one for analysis also, but for this purpose the design object will be shown). Under the design object are the fuselage and wing objects. The substructures can be found under the fuselage, and the wing number and the airfoils of the wings are located under the wing. The AMRaven toolbar was used to create all of the aforementioned objects. Once again, this makes modeling faster and easier. Instead of having to draw and create an airfoil, the user can modify the airfoil object in the Model Tree to the desired size. The same steps

could be done for changing substructures within the fuselage. Rather than having to create every single substructure by lines, curves, and surfaces, the user can change the substructure quantity property, size property, and any other variable that needs to be altered.

2.2.3 Building a Wing

Building a wing becomes an easy task when using the AVB. A novice or senior designer will find it simple to use, and quick for baseline modeling of a wing. Plus, the GUI allows for quick learning of the system and easy modeling without the need to write code. A wing-building toolbox called the “Wing Editor Form” (WEF) is available within the AVB and contains all of the needed objects to design a wing. The WEF is self-explanatory and contains variable descriptions for the work in progress and a number of tabs to complete many tasks. The first tab is the “Configuration” tab. It has many different variables that affect the layout and structure of the wing and is illustrated by Figure 2-7. One of the important features on this tab is the ability to change the number of wing sections on the wing. The user can select any amount of wing sections for the design so that different material properties, sweep angles, or any other variables can be applied to each section.

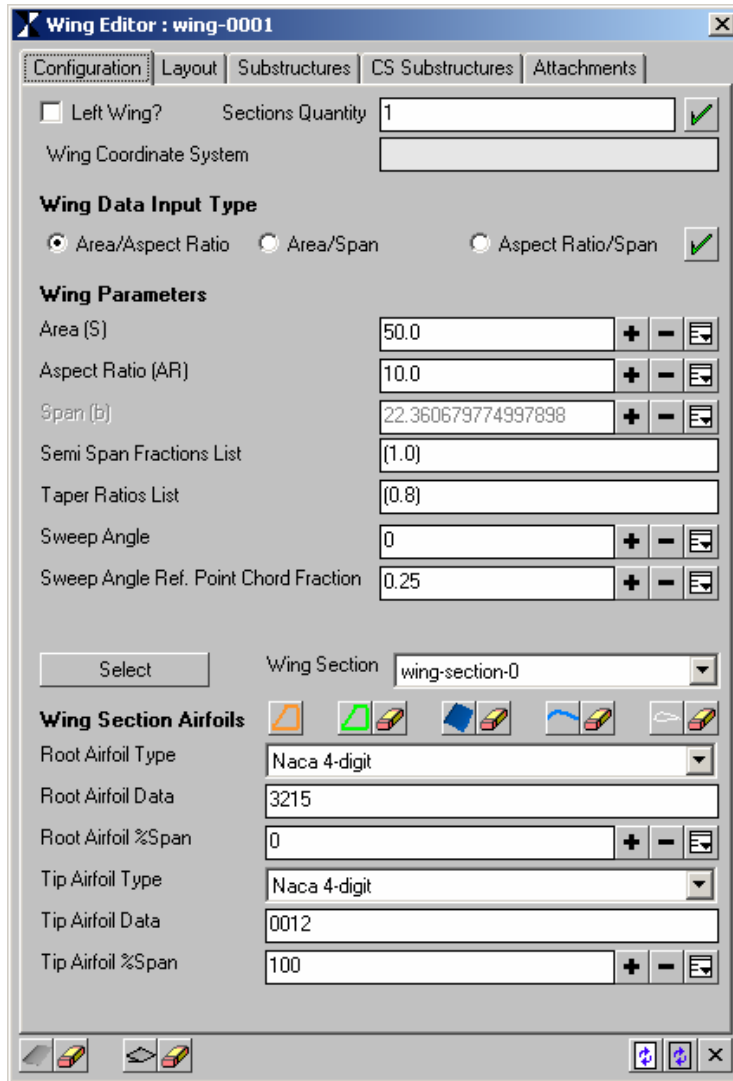


Figure 2-7: Configuration Tab

The second tab on the form is the “Layout” tab. This tab is the main tab for changing the layout of the wing. As you can see on this tab, variables such as incidence angle, sweep angle, twist angle, and many more are available for changing the layout of the wing and wing sections.

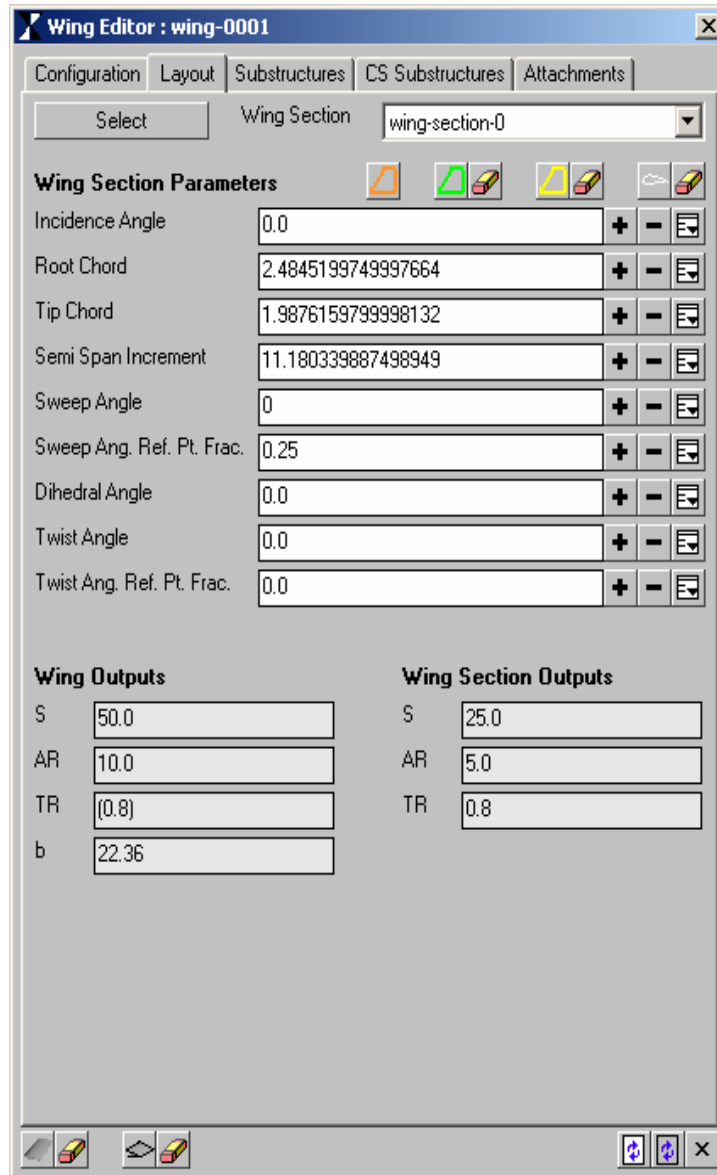


Figure 2-8: Layout Tab

The third tab is the “Substructures” tab that enables the designer to add substructures such as ribs, spars, and control surfaces to the wing. A grid-point layout button is on the tab that will draw grid points on the wing planform so that the substructures can be added to the wing in the proper locations. These grid points can be spaced however the user desires. Figure 2-9 illustrates the “Substructures” tab.

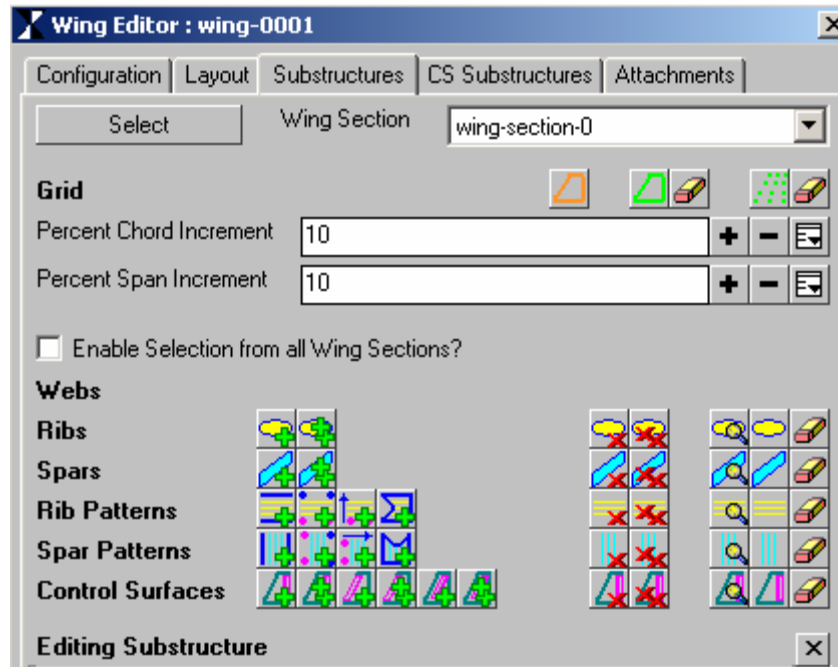


Figure 2-9: Substructures Tab

The next tab is the “CS Substructures” tab that allows the engineer to add substructures to the control surfaces. This tab is used in the same way as the “Substructures” tab, only it is for substructures on the control surfaces. Figure 2-10 shows this tab.

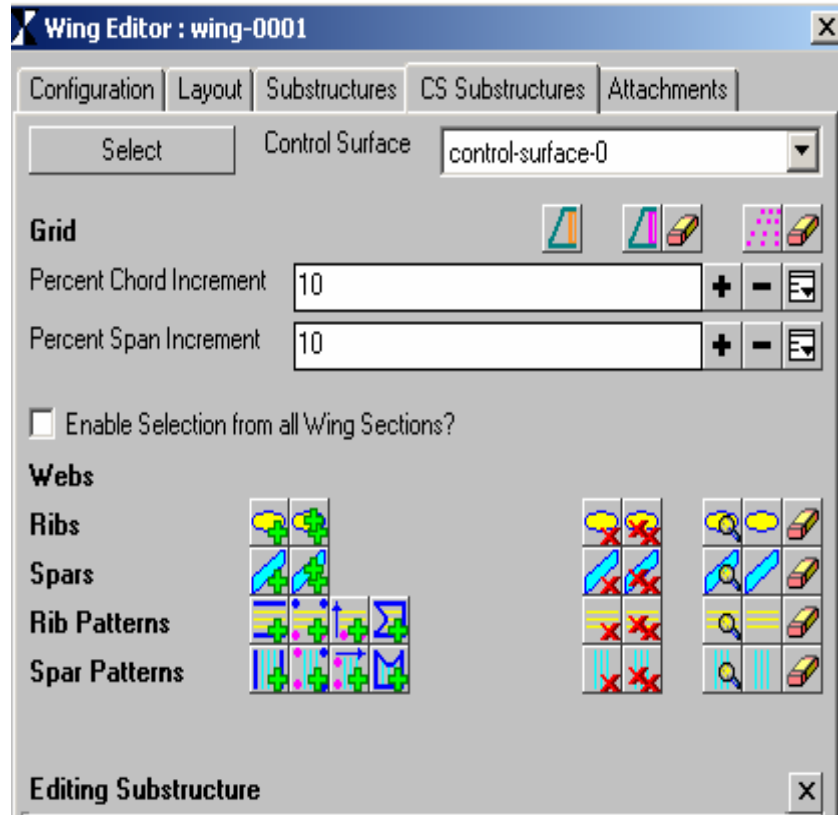


Figure 2-10: CS Substructures Tab

The last tab on this form is the “Attachments” tab. This enables the user to add hinges, hinge lines, and support trusses that help with finite element analysis.

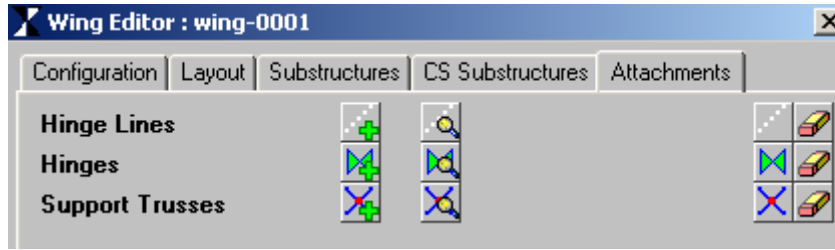


Figure 2-11: Attachments Tab

Each of these tabs is explained in more detail in Chapter 3 in the AVB user manual located in Appendix A. Furthermore, there is also a case study for modeling a wing within the manual.

2.2.4 Building a Fuselage

The fuselage builder within the AVB is a straightforward tool that enables the user to designate the fuselage geometry of the aircraft being drawn. It allows for the selection of any shape or configuration of a fuselage. Just like the wing builder, it does not require the user to know background AML programming or to write code. The fuselage builder toolbox does not help in drawing the external shape and configuration of the fuselage. Due to the different possible fuselage configurations, the user must execute other geometric objects within the AVB such as curves, boxes, and many more to draw the shape of the fuselage. Once the external shape of the fuselage is drawn, the fuselage toolbox then allows the selection of this shape for the fuselage and then aids in the design of the internal structure of the fuselage. The fuselage builder helps to draw bulkheads, walls, floors, and frames in the fuselage.

The fuselage builder GUI called the Pod Editor contains two tabs: 1. Configuration; and 2. Substructures. Figure 2-12 expresses the “Configuration” tab, and Figure 2-13 shows the “Substructures” tab. The “Configuration” tab is used to select the external shape of the fuselage and to decide how the object would be meshed if finite element analysis was completed. The “Substructures” tab enables the user to add bulkheads, floors, frames, and walls to the structure.

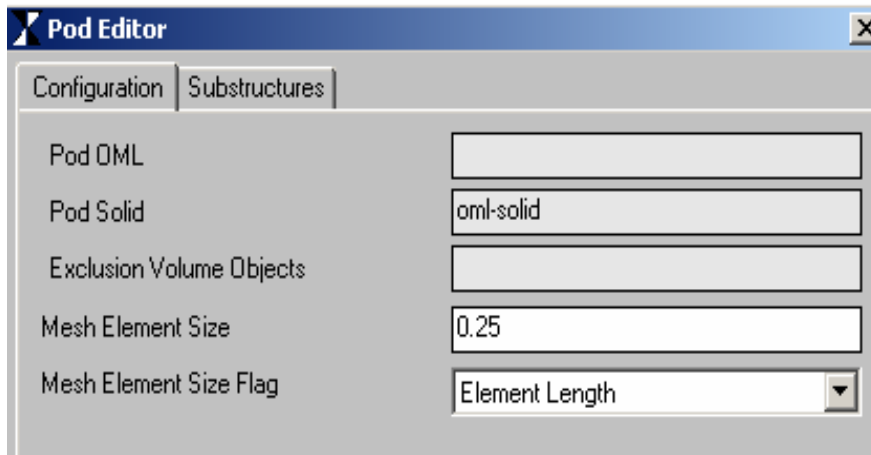


Figure 2-12: Fuselage Configuration Tab

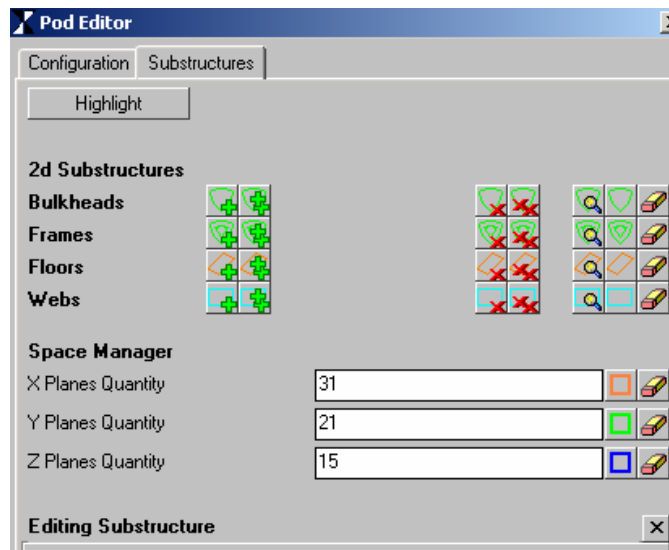


Figure 2-13: Fuselage Substructures Tab

2.2.4.1 Drawing the Fuselage

As stated above, the Pod Editor tool allows for the selection of the fuselage structure and the design of the internal substructures. However, the external shape of the fuselage must be modeled using other objects within the AVB. Once again, this process is not difficult as the objects are available within the AVB, easily obtained, and only need to be modified to the desired shape rather than written with AML programming from scratch.

There are many geometric objects available in the AVB when drawing a fuselage shape. There are 0D, 1D, 2D, and 3D primitives. The 0D primitives contain point objects. The 1D primitives contain lines, circles, arcs, vectors, and curves. The 2D primitives contain sheets, discs, open elbows, open cones, open cylinders, and surfaces. The 3D primitives contain boxes, cones, cylinders, pipes, and many more. Each of these can be accessed by using the tool bar within the AVB. There is a pull down menu for each as shown in Figure 2-14.



Figure 2-14: Primitive Menu Options

Furthermore, there are airfoil objects available if the aircraft has a fuselage shaped similar to an airfoil, such as a blended-wing-body airplane. The objective is to use the various objects available and to combine them to draw the fuselage geometry. Chapter 4 in Appendix A contains more information on how to use the fuselage toolbox, along with an example of building a fuselage model.

2.2.5 Other Tools

Presently, there are no toolboxes for items such as tails, fuel tanks, cargo, landing gear, and pylons. However, similar to drawing a fuselage, the AVB includes tools that will allow the user to model these other components of the aircraft. For example, the wing builder toolbox can be used to draw a tail. The coordinate system can be changed so that the tail is in the proper position, and the tail can be drawn just like a wing. Or, landing gear could be drawn using objects similar to those used to draw the fuselage. Structural and mass properties could be given to these objects. Once the other structural components are drawn, the AVB allows simple connection of these objects.

2.2.5.1 Connecting Structures to the Aircraft

Objects such as landing gear, wheels, cargo, and any other structural components, can be connected to the aircraft model in the AVB very easily. The AVB has objects in it that allow for connection in many different ways. For example, the GUI contains buttons that aid in connecting the wing to the fuselage. Selecting the button and then choosing what substructures to attach together can do this. Furthermore, the AVB has a button that the user can select to attach other substructures to the airplane, such as landing gear, fuel tanks, or any other structure. Once again, the only task in this is selecting the structures that need to be connected. Another object within the AVB is called a “sewn-object.” This object allows the user to “sew” together any lines that needed to be connected to make a desired shape. The advantage of this is that instead of having to draw the model line-by-line in order, the user can draw individual components and connect them later.

3. INTEGRATION

3.1 Integration Background

Learning, using, and integrating the tools used to design aircraft can be a difficult task. However, the AVB promotes an environment that makes this process simpler. If a desired procedure or formula not available within the AVB, then the AVB integrator is set up to allow a user to incorporate his/her own code in order to use the desired function without much work. It also can output the modeled aircraft into other CAD programs like IDEAS. Furthermore, the AVB is set up to integrate finite element modules like NASTRAN with the AVB model created. However, the finite element code will be discussed in Chapter 4 of this study because its own content.

3.2 External Programs

If a desired procedure or formula is not available within the AVB, then the AVB integrator is set up to allow a user to incorporate his/her own code in order to use the needed formula. There is not a limit on what program can be integrated with the AVB because the AVB is set up to communicate with any program. The AVB contains procedures within it that can write an input file for an external program, or read an output file from an external program to process within the AVB. In other words, the AVB can write an input file in the correct format so that the user does not have to write any code. The advantage of this is that the designer does not have to write long code to integrate two programs.

3.2.1 Integration Example

One program used by many designers is MATLAB. MATLAB contains numerous value functions that calculate many equations to solve for eigenvalues, matrix operations, and many more. Furthermore, after calculating these values, MATLAB can write a text file that can be used by another program. The AVB can start MATLAB, run

the MATLAB code, and read any output file data for processing. The purpose of this example is to express the capability of integration within the AVB and its simplicity.

3.2.1.1 Integration Scenario

The goal of this example is to calculate the surface area of a wing planform. A wing will be modeled within the AVB with specific chord lengths at the root and tip airfoils. The two values, as well as the span length of the wing will be outputted to a text file that will be read by MATLAB. MATLAB will take the values and calculate the surface area of the wing. The surface area value will be written to a text file that will be read and processed by the AVB. The whole process will be started in the AVB environment. Take note that surface area calculation is already done within the AVB, but is being used for the purpose of this example.

3.2.1.2 Integration Environment

The AVB environment contains the chord values and span length of a wing. These variables fall under a certain object within the Model Tree, and can be referenced accordingly. Figure 3-1 illustrates the location of the chord length variables within the Model Tree hierarchy. Furthermore, the work area section shows the span length variable and how it falls under the construction object.

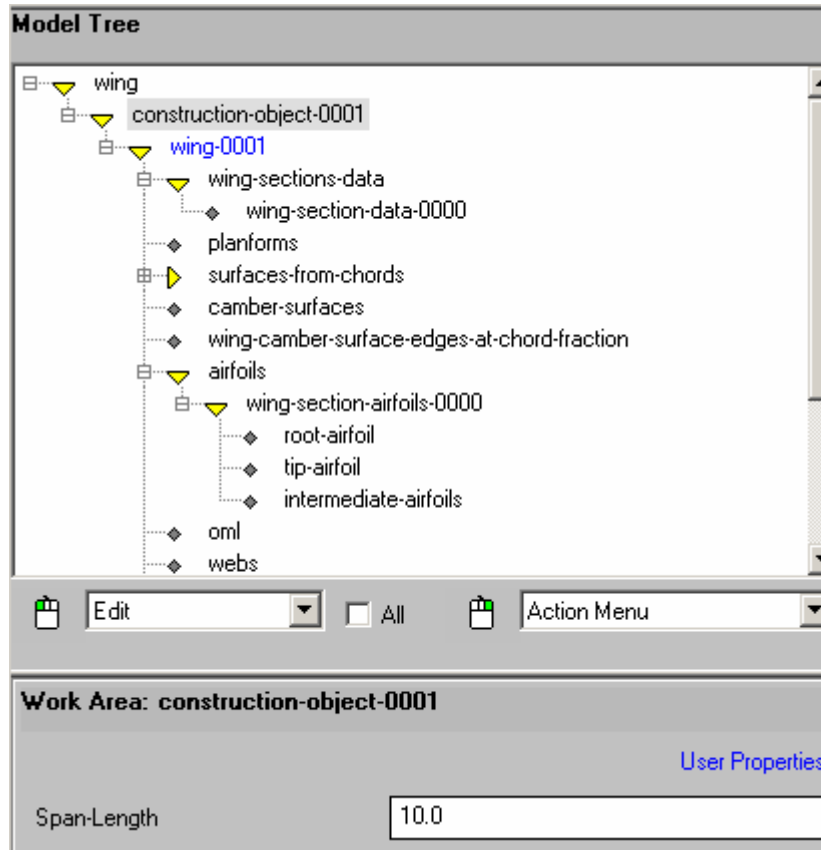


Figure 3-1: Wing Model Tree

3.2.1.2.1 Properties-file-io.aml File

The AVB contains a procedure called “properties-file-io.aml.” This file has code that will write and read the variable information into and from a text file. All the user has to do is edit the code to reference the proper variable names and locations in the Model Tree and the code will do the rest. The “properties-file-io.aml” file is located in Appendix B.

The “properties-file-io.aml” file within the AVB is written to reference a generic object. Thus, the designer can copy the code to another file, choose a name, and change the referencing to the proper object. There are write and read property object list sections within the file that must be copied to the new code in order for it to read properly. These

functions are called “write-property-object-list-to-file” and “read-property-object-list-to-file” respectfully.

3.2.1.2.2 Properties List

The following portion of code from the “properties-file-io.aml” file gives an illustration of the property list within the code. The property list is the list of variables that will be passed back and forth between programs.

```
(defun my-file-io-write-test ()  
  (let* (  
    (property-objects-list '(  
      (the wing construction-object-0001 Fuselage-Length)  
      (the wing construction-object-0001 Fuselage-Span)  
    ))  
  )
```

The user would edit the “property-objects-list” to refer to the desired variables. For this example, the property object list would look like the following.

```
(the wing construction-object-0001 wing-0001 airfoils root-airfoil chord-length)
```

```
(the wing construction-object-0001 wing-0001 airfoils tip-airfoil chord-length)
```

```
(the wing construction-object-0001 wing-0001 wing-section-data-0000 semi-span-  
increment)
```

```
(the wing construction-object-0002 Surface-Area)
```

3.2.1.2.3 Writing to a Text File

The following portion of code from the “properties-file-io.aml” file gives an illustration of the code that writes the variables to a text file.

```
(write-property-objects-list-to-file property-objects-list
"c:\\temp\\my-file-io-write-test.txt")
)
) ;; End of function my-file-io-write-test ().
```

The “write-property-objects-list-to-file” portion of the code contains the location that the text file will be written to, and the name of the text file. Both items can be altered to the desired location and name.

3.2.1.2.4 Reading a Text File

The following portion of code from the “properties-file-io.aml” file gives an illustration of the code that reads the variables to a text file:

```
(defun my-file-io-read-test ()
  (read-property-objects-list-from-file "c:\\temp\\my-file-io-write-
test.txt")
) ;; End of function my-file-io-write-test ().
```

Just like with the write section, the location and text file name can be changed to the correct file that will be read. Whatever file is chosen, the property values will be read from that file. The name of the file should be the name of the text file outputted by MATLAB.

3.2.1.2.5 Batch File

The AVB has the capability to call a batch file in DOS. The usefulness of this is that MATLAB can be started and ran using a batch file. Thus, a batch file will be created by the user, and then used to call MATLAB and run the appropriate code. The batch file

for this example is located in Appendix B. The only thing required by the AVB is the command to call the batch file that runs MATLAB. The command is:

```
(run-program "C:\\temp\\integration.bat" :windows-batch-file? t)
```

The batch file can be placed in the desired directory of the user, as long as the user references the proper directory in the command line. The batch file command line to run a Matlab script is as follows:

```
"matlab -nosplash -nodesktop -minimize -r integration"
```

The "nosplash," "nodesktop," and "minimize" commands run the MATLAB script with a minimized window so that the MATLAB program window does not interfere with the AVB window. The "-r integration" part of the line is calling the correct m-file, which is the title the user gave it. The only other task that must be completed before running this command line is that the directory must be changed to where the MATLAB script is.

3.2.1.2.6 Emacs Command Window

There is a command window available with the AVB system called "Emacs." Figure 3-2 illustrates this window. This window is where all the commands will be written for this process. In other words, the batch file run command would be typed in this window and run.

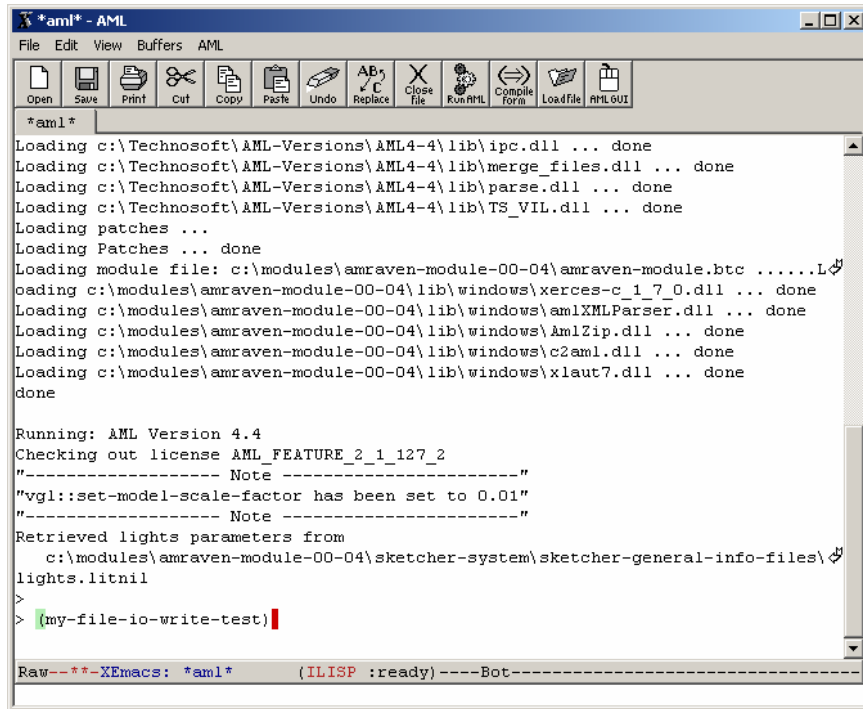


Figure 3-2: Emacs Window

3.2.1.2.7 Integration Procedure

The following is a step-by-step procedure on how to export the chord and span length variables to MATLAB, run the desired code in MATLAB, and post process the information in the AVB. It is assumed that the batch file to run the MATLAB has already been created, as explained in the “Batch File” section 3.3.1.2.5.

1. Copy the “write-property-object-list-to-file,” “read-property-object-list-to-file,” and the sections that write and read the list to a specific location. Paste this code into a new file and save it as a desired name. For this purpose, call it “integration.aml.”
2. Change the property list to the list given above in section 3.3.1.2.2.
3. The default name for the text file written will be “my-file-io-write-test.” The default location of the file is “C:\\temp.” If this name and location does not please the user, it can be changed to the proper name and location. For this example, the default name and directory will remain the same.

4. The default name for the text file to be read back into the AVB is “my-file-io-read-test.” Once again, this can be changed, and for this example it is called “integration.txt.”
5. Make sure the default directory has been created with read and write privileges.
6. Type (my-file-io-write-test) in the command window and press “Enter.”
7. Now run the batch file that calls and operates the MATLAB script.
8. The MATLAB file calculates the surface area of the wing planform and writes it to a text file called “integration.txt.” Now, the user must read the surface area value back into the AVB. Type "(my-file-io-read-test)" in the command line and press “Enter.”
9. Finally, maximize the AVB sketcher window and hit the “regen” button in the work area. The surface area value will be changed to the calculated value from the MATLAB code.

3.2.1.2.8 Integration Run

The batch file was run and yielded the proper results. The chord lengths had values of ? and ? which produced a surface area of ????. Figure 3-8 displays the Work Area with the calculated results.

3.2.1.2.9 Integration Usefulness

The usefulness of this integration process is clear. Instead of the user having to start MATLAB, enter values of the variables, and calculate the surface area, the user only has to type a few commands in the command line, and the AVB will do the work for the user. This saves the time and energy devoted to learning multiple programs and having to write long code in order to integrate selected programs. Furthermore, the exterior programs can calculate values that can be passed to the AVB and used to alter the design of the model, proving even more that the AVB is a great integrator.

3.3 CAD Programs

Once the aircraft is drawn in the AVB sketcher it can be exported into various CAD program formats. For example, if the designer was more comfortable using IDEAS as a solver than the AVB, the AVB can create an IDEAS model of the same aircraft. The options include exporting the file as an IGES, DXF, or STEP file.

3.3.1 Exporting Example

This example is used to show the ability of the AVB to export a model into other CAD program format. The goal is to export a wing so that it can be viewed and edited in IDEAS. A simple trapezoidal wing is used as shown in Figure 3-3.

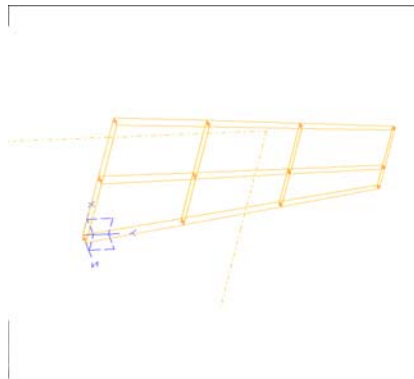


Figure 3-3: IDEAS Wing Model

3.3.1.1 Exporting A Model

Once the model is created in the AVB, exporting it is very easy. All the user needs to do is use the AML main menu toolbar displayed in Figure 3-4 and select the “Model/Save/IGES Geometry” option. The AVB will prompt the user to name the file, and then write the model into IGES format. The created file can then be opened up and viewed in IDEAS. Figure 3-3 illustrates the IDEAS model. Once in IDEAS the user has the option to edit the model and save it in IDEAS.

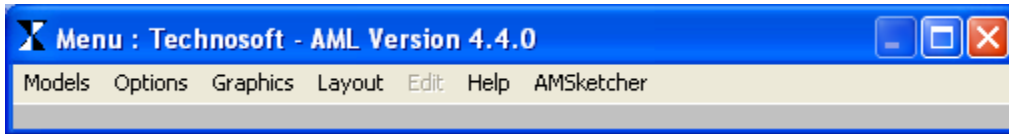


Figure 3-4: AML Menu Toolbar

3.3.3 CAD Program Usefulness

The ability to export the AVB model into other CAD programs is very useful. It allows a user that wants to use another program to edit the model to do so. Furthermore, if another engineer needs to view the model, but does not have the AVB program on their computer, the AVB can export the file into the proper format of a program that the other engineer does own. This is nice for concurrent engineering purposes and integration.

4. FINITE ELEMENT ANALYSIS

4.1 Finite Element Analysis

Finite element analysis is another important process in aircraft design. After a baseline model is created the engineer must evaluate if the design is stable. Thus, the finite element input file of the model is created so that it can be assessed in a finite element solver like NASTRAN. Input files can take a lot of time and be tedious to produce. Making sure all the proper input cards are created alone can take many hours. However, the AVB integrator has the capability of integrating with various finite element programs so that the user has to do very little work. It is capable of creating the proper input file for analysis without the user having to enter any input lines. Furthermore, it can run the input file in the solver and then read the output back into the AVB and update the model. This tool is very useful as it saves the designer time from having to create the finite element input file and allows the user to integrate with the finite element solver in order to use the output file. The output file can be used to gather information like displacements and stress and use them as constraints in optimization or create contour plots of the results. All this is done by the AVB without the user having to learn how to create the input files or read them. This chapter describes the capabilities of the AVB finite element integrator.

4.1.1 Finite Element Tools

There are many different input options when making a finite element model. However, there is common input cards used in almost every finite element model of aircraft. Thus, the AVB contains buttons that help to create these input cards in the input deck of the NASTRAN file. These cards include substructure attachments, constraints, support cards, point loads, and point masses. The buttons are located on the AMRaven Toolbar, and their descriptions will be displayed when holding the mouse pointer over the button. This gives fast access to creating constraints, loads, masses, support cards, etc., for the NASTRAN input deck. Instead of the user having to type in the line-by-line commands, the AVB knows to create the proper cards for the input deck. This alone

saves the user a great deal of time. In essence, the user does not need to know what the input card line should be, thus saving the time of having to learn the proper input format. Chapter 5 in the user manual attached in Appendix A gives more details of these button options. Table 4-1 gives a list of the finite element cards on the AMRaven toolbar and their NASTRAN equivalents.

Table 4-1: AMRaven NASTRAN Cards

AMRaven Tool	NASTRAN INPUT CARD
<i>Substructure Attachments</i>	RBAR
<i>Constraints</i>	SPC, MPC
<i>Support Cards</i>	SUPPORT
<i>Point Load</i>	FORCE
<i>Point Masses</i>	CONM2

4.1.2 Finite Element Options

The AVB provides the designer with various finite element analysis options. There is a pull down menu as shown in Figure 4-1 that displays all the selections. The user only needs to select the type of analysis desired to be completed, and the AVB takes care of the rest! For example, just like with t of analysis to run--once again saving the engineer time. Furthermore, the designer do he input cards, the AVB will also write the proper start deck and tell NASTRAN what type es not have to learn what the proper input numbers are required in order to run the correct analysis, which again proves the AVB is a great time saver and integrator!

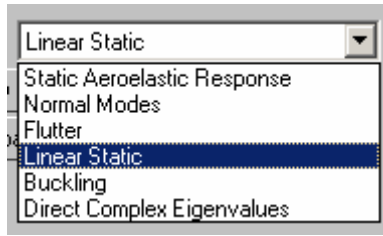


Figure 4-1: Analysis Types

4.1.3 Aerodynamic Analysis

Aerodynamic analysis can also be completed by the AVB. An aero-panel model is designed in the AVB and set up as the designer desires. The AVB automatically creates an aerodynamic analysis object so the user only has to edit it. Once made, the AVB can create the needed input cards for the NASTRAN input file. The user does not have to write any input lines for the file.

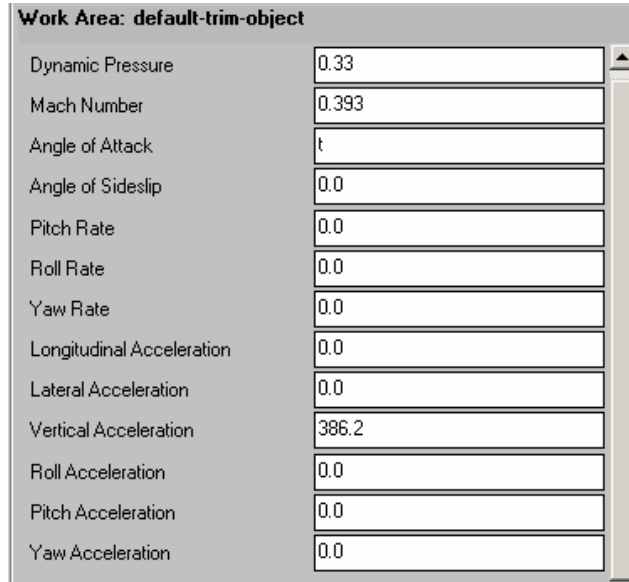
4.1.3.1 Aerodynamic Panel Model

One of the first items required when completing aerodynamic analysis is an aerodynamic panel model. When a wing is created in the AVB an aerodynamic panel model object is automatically created. One advantage of this is that the designer does little extra work in creating the panel model. The designer only has to set the number of desired aerodynamic panels if the default is not good enough. Once again, the AVB will know how to create the proper input cards for the NASTRAN file so that the user does not have to enter this information into the file.

4.1.3.2 Aerodynamic Properties

Once an aerodynamic panel model is made, aerodynamic properties such as Mach number, dynamic pressure, and others can also be edited very easily in the AVB. The user has the option to change the values of these properties to the desired value and then update the model. These values will be placed into the NASTRAN analysis by the AVB,

saving the designer time and helping with the integration. The aerodynamic property list is shown below in Figure 4-2.



The screenshot shows a software window titled "Work Area: default-trim-object". It contains a list of aerodynamic properties, each with a corresponding input field. The properties and their values are as follows:

Property	Value
Dynamic Pressure	0.33
Mach Number	0.393
Angle of Attack	t
Angle of Sideslip	0.0
Pitch Rate	0.0
Roll Rate	0.0
Yaw Rate	0.0
Longitudinal Acceleration	0.0
Lateral Acceleration	0.0
Vertical Acceleration	386.2
Roll Acceleration	0.0
Pitch Acceleration	0.0
Yaw Acceleration	0.0

Figure 4-2: Aerodynamic Properties

4.1.4 Meshing

Naturally, before running a model in a finite element program like NASTRAN a mesh of the model must be generated to produce elements. The meshing is done by PATRAN. The AVB has the option to create many different types of elements depending on what type of aircraft object is created. For example, CSHEAR or CSHELL elements can be used to represent spars and ribs, while quadrilateral elements (CQUAD4) are use to represent the wing skin. Appendix B contains a NASTRAN input file of a trapezoid wing study in this research, illustrating the different elements created by the AVB.

4.1.4.1 Meshing Procedure

Meshing is very easy when using the AVB. Once the model that is being meshed is created, the user only needs to draw the mesh. There are two options: 1. “Individual Component Mesh,” or 2. Full Model Mesh.” The first step in using the “Individual Component Mesh” is to follow the path shown in Figure 4-3.

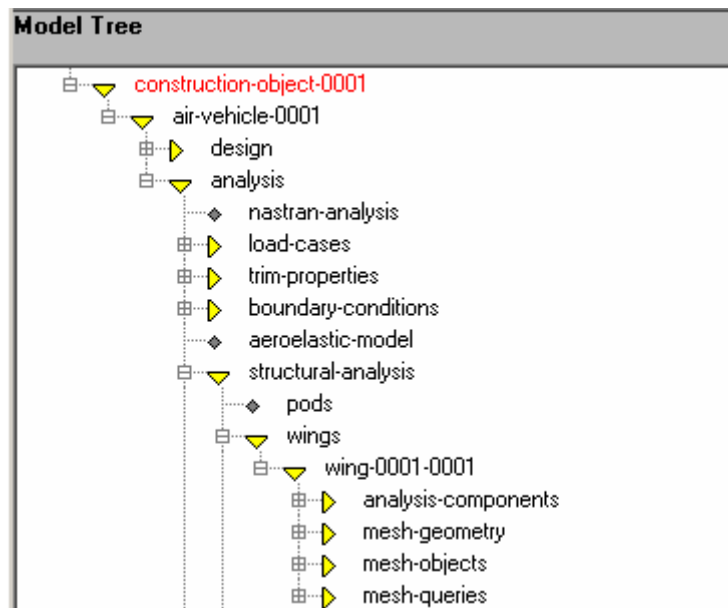


Figure 4-3: Meshing Path

Once the user gets to the “mesh-object” path, you can select whichever object you wish to mesh, and use the “draw” option. The AVB will use PATRAN to mesh the object and to draw the mesh in the sketcher window.

The full model mesh can be drawn using the “post-processing” tab in the Model Tree and by selecting to draw the “Element-Query.” Figure 4-4 provides a picture of a mesh of a wing using the full model mesh option.

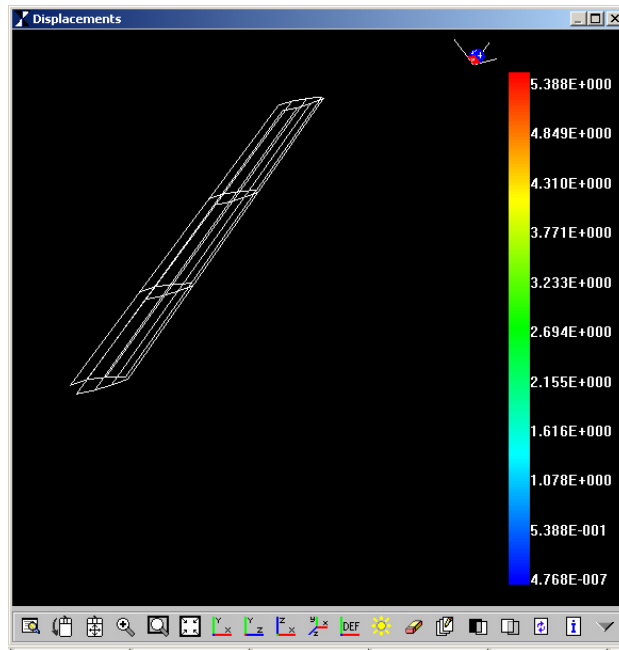


Figure 4-4: Wing Mesh

4.1.4.2 Making The Mesh Finer

The AVB has an option to allow the user to make the mesh finer or less fine. It is available with a variable called “mesh-object-size-across-web-caption.” which is located under the “Analysis/design/wings/wing-0001” in the Model Tree. The value of the variable effects the number of elements that run across the structure root-wise. Figure 4-5 illustrates a wing with the “mesh-object-size-across-web-caption” being 3, while Figure 4-6 displays the mesh with it being equal to 6. The value of the variable times two is the number of elements that will be across the element chord-wise.

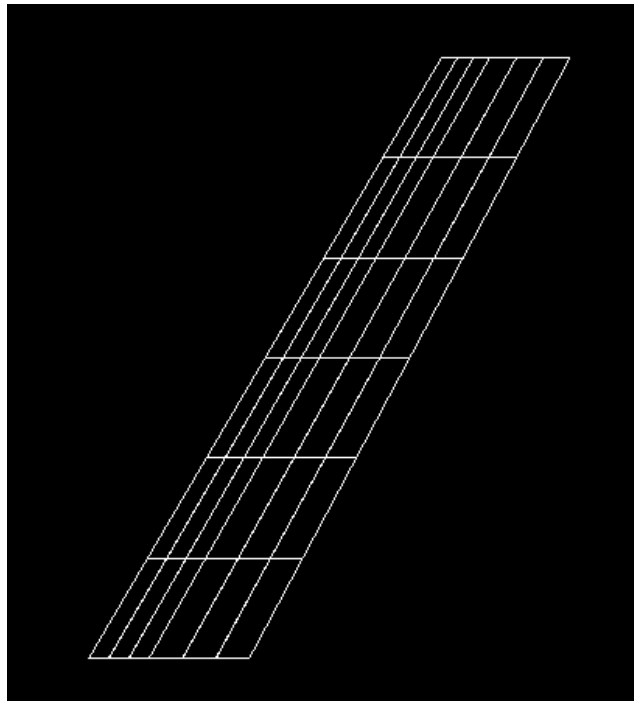


Figure 4-5: Wing Mesh Case One

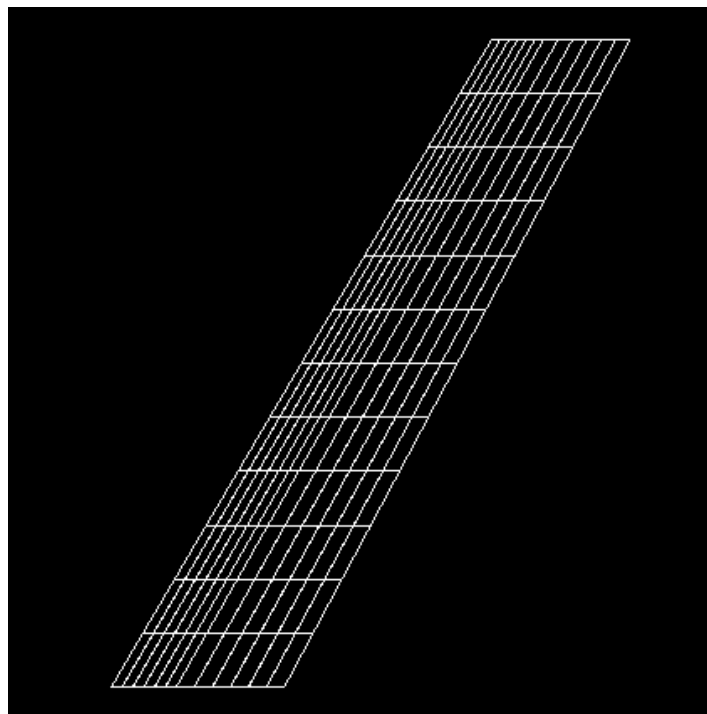


Figure 4-6: Wing Mesh Case Two

4.1.5 Finite Element and Aerodynamic Integration

Once a model is drawn and the proper conditions, loads, analysis, etc., are set up the aerodynamic and/or finite element analysis can be run. Integrating the AVB with the needed programs (i.e. PATRAN and NASTRAN) is very simple. The AVB comes with the needed components to write and read the proper input files. The only thing the user has to do is to enter the proper command lines in the “logical.pth” file mentioned in Chapter 2 of the user manual in Appendix A. There are four lines needed as shown below.

```
;; MODIFY ACCORDING TO YOUR PATH AND VERSION:
:patran-path          c:\Msc.Software\Msc.Patran\2004r2\
:nastran-path        c:\msc\bin\
:meshes              c:\temp\meshes\
:nastran-data        c:\temp\analysis\
;;
```

If the user loaded the AVB according to the instructions given in the manual, then these lines are already in the “logical.pth” file. Otherwise, the user will need to enter them. The objective is to reference where the location of the PATRAN and NASTRAN home directories are on the modeler’s computer, and to reference two folders to hold the mesh and analysis data files.

Once these command lines are placed in the “logical.pth” file, the user can easily run aerodynamic or structural analysis. He or she only needs to go to the “post-processing” object in the Model Tree and click to “draw” the displacements. This will trigger the AVB to run the proper analysis (aerodynamic and structural) in order to retrieve the displacements. The simplicity of this alone makes the AVB useful. A great deal of time has been spent on trying to integrate programs; however, the AVB does the integration for the user.

4.1.6 Reading and Post-processing Files

The input and output files can be read once analysis is set up or completed. The input files can be found in the “meshes” and “analysis” folders created. The “meshes” folder contains all the mesh data files, and the “analysis” folder contains all the input and output NASTRAN files. If desired, the user can open up these files and read them from this location.

The AVB also has the capability to read the files. For example, once the analysis is run, the AVB can import the results (i.e. deformation, stress, frequencies, etc) and use them to change the model to a new configuration or display the deformed model. The user only needs to select what type of post-processing is desired (i.e. displacement or contour plots), and select to “draw” the results. Figure 4-7 illustrates a deformation plot of a wing.

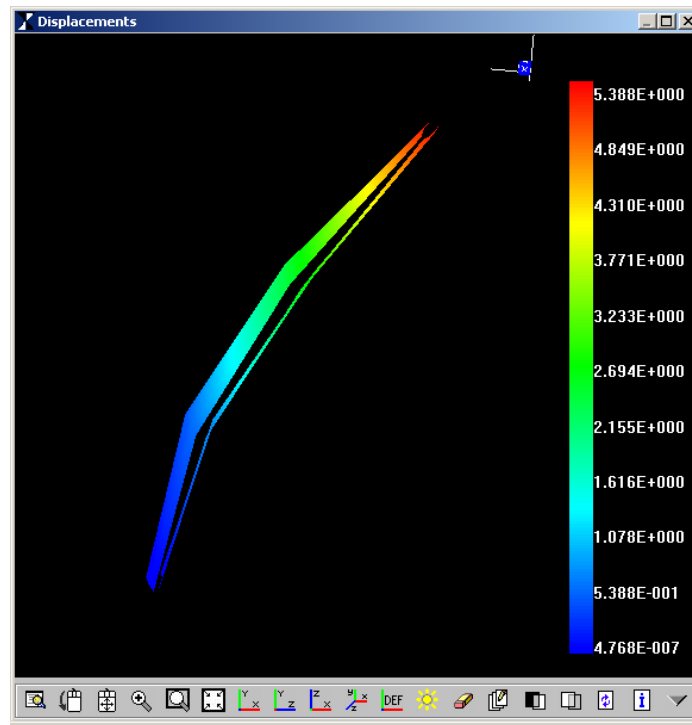


Figure 4-7: Deformation Plot

This utility is nice because it allows concurrent engineering to take place when designing the aircraft. In other words, the user can use NASTRAN results to change or optimize the model. The best part is the AVB integrates the whole process for the user without the designer having to script any of the code. The post-processing capabilities will be illustrated in Chapter 5 when a wing is optimized.

4.1.7 Finite Element Usefulness

The AVB is set up to integrate with NASTRAN and makes finite element analysis easy. Once a model is created in the AVB, the user can add constraints, loads, support cards, and many more to model and the AVB will create the input cards for a NASTRAN input file. Furthermore, the AVB can run and post-process aerodynamic, structural, aeroelastic, and dynamic analysis. It can read the results and use it to edit or optimize the model. The user does not have to code any input which makes it nice, because if someone is not familiar with NASTRAN and its input cards, the AVB will create the necessary cards. The user does not have to spend a lot of time creating and understanding the proper cards. Plus, this enables the ability to optimize a aircraft by a click of a button and allowing the computer to do all the work. The AVB is a great finite element tool.

5. OPTIMIZATION

5.1 Optimization Introduction

Designing and building an airplane is a costly process. Thus, anywhere money can be saved is a goal. Therefore, optimizing the shape and size of an aircraft to minimize the weight is an important step in the design process. The Air Vehicle Builder (AVB) aids in optimization in two ways: 1. Outputting a NASTRAN file, or 2. Using the AMOPT toolbox to complete an optimization study.

5.2 NASTRAN File

The AVB has the capability of creating an optimization file for NASTRAN. Once a model is created with proper loads and boundary conditions, the user only needs to select the optimization option and the NASTRAN file will be created. This is done by going to the “analysis” portion of the Model Tree, and using the “load-cases/default-load-case” option as shown in Figure 5-1.

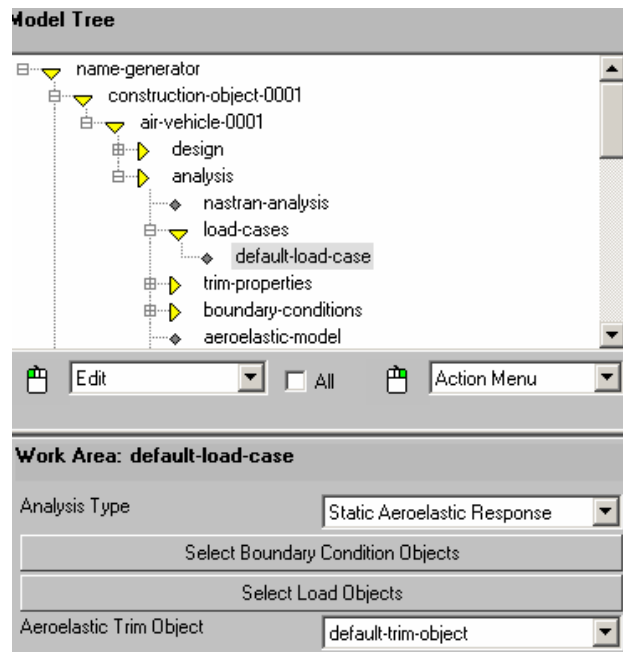


Figure 5-1: NASTRAN File Creation

5.3 Optimization Environment

The AVB is also set up as a great optimization environment. It employs a toolbox called AMOPT that contains many valuable tools to make the optimization process easy. Its applications include Sensitivity Analysis, Design of Experiments (DOE), and a DOT optimization tool with the feasible directions method to optimize a design. These methods save the user time and energy because the AVB does the most work. The user only needs to create design variables, design constraints, and the model while the AVB sets up and runs all the analysis.

5.3.1 AMOPT Toolbox

The AMOPT toolbox allows the user to complete a full optimization study of an aircraft. It is very user friendly and easy to learn.

5.3.1.1 AMOPT Toolbar

Figure 5-3 shows the toolbar for AMOPT. The AMOPT toolbar contains various push buttons to quickly create the needed objects for the optimization study. Presently, the “Add an AMOPT exploration/optimization object” button is the only working button. Technosoft plans on writing the code for the other buttons in the future. By selecting the “Add an AMOPT exploration/optimization object” button the needed optimization object will show up in the model tree to run the optimization study



Figure 5-3: AMOPT Toolbar

5.3.1.2 Optimization Object

Once the optimization object is created the engineer can edit its features. Figure 5-4 displays the “Exploration/optimization Object” with all the optimization options.

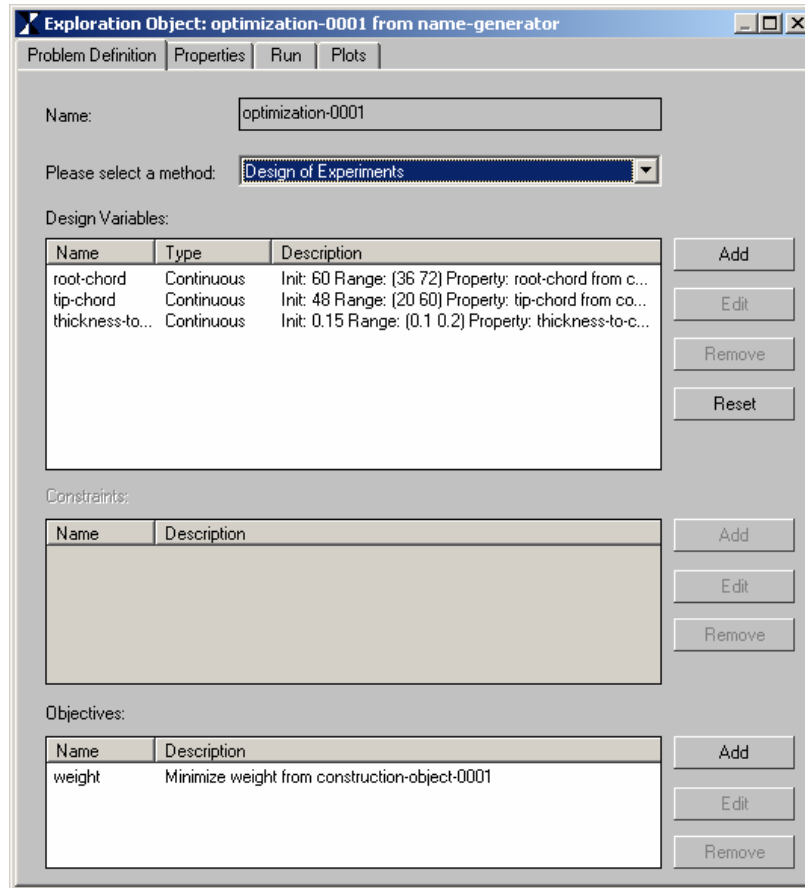


Figure 5-4: Exploration Toolbox

The designer can choose from a variety of different methods to perform the study. These options include DOE, DOT, Parametric Study, Monte Carlo, Multi-Objective GA, Nelder-Mead, Powell, and SQP. Once the method of optimization is chosen the user can begin adding design variables, constraints, and objectives by hitting the “Add” button by each item. The “Design Variable” window is shown in Figure 5-5. The user selects the variable from the tree or graphics and then gives the min and max value.

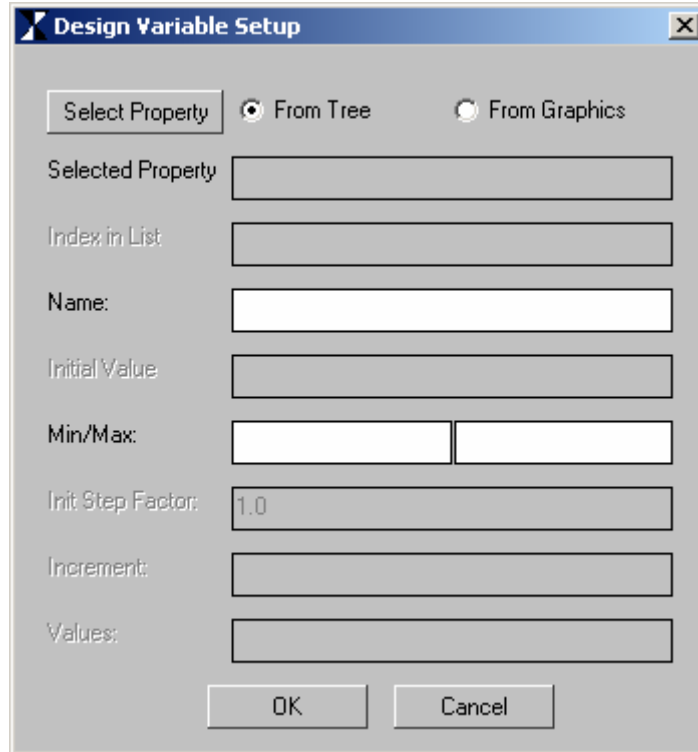


Figure 5-5: Design Variable Window

The “Constraint Setup” is very similar to the “Design Variable Setup.” The designer adds the constraint from the tree or graphics, selects the relationship, and gives the constraint value. Figure 5-6 displays the “Constraint” window.

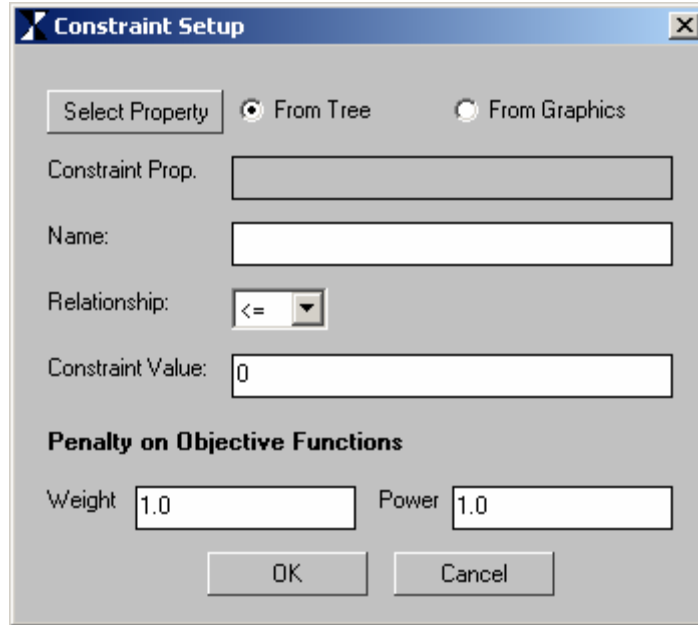


Figure 5-6: Constraint Window

The objective function can be created many ways in the AVB. It can be a formula, variable, or call an external program to evaluate a function. The user only needs to create a variable to represent the formula, variable, or external program call so that AMOPT can reference it as the objective.

The “Objective Setup” is used by selecting the objective function, and choosing to maximize or minimize its value. The objective can be a variable that is calculated by a function, some property within the AVB, or call another code that calculates the needed value. Figure 5-7 illustrates the “Objective” window.

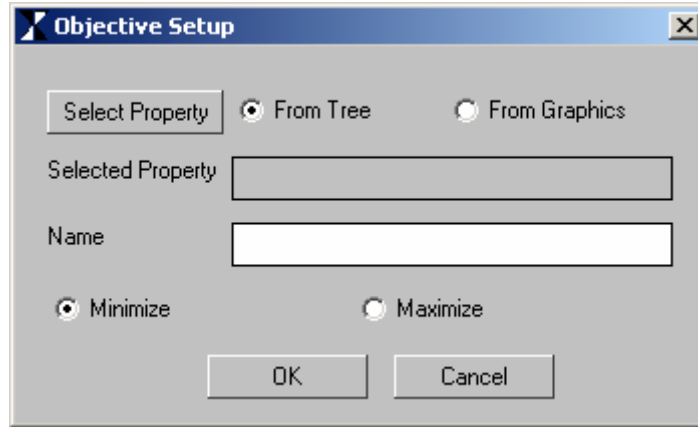


Figure 5-7: Objective Window

After the design variables, constraints, and objective functions are set up, the user can change to the “Properties” tab on the “Exploration Object” window to set up the properties of the optimization run. The “Properties” tab can vary based on what type of optimization method has been chosen. Figure 5-8 shows the the “Properties” tab and its options for the “DOT Optimization” method. The tab allows the designer to choose method type, output filename, optimization step size, and many more options.

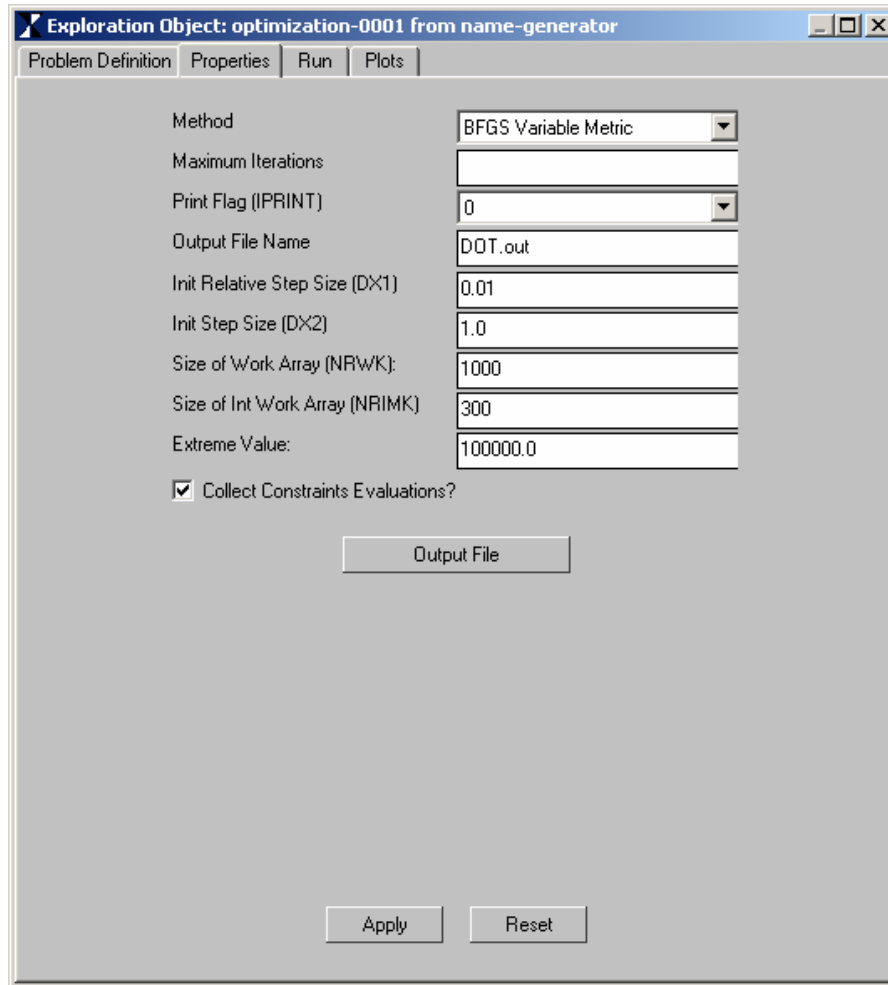


Figure 5-8: Properties Tab

Next, the “Run” tab can be used to run and save the optimization results. Figure 5-9 displays the “Run” tab.

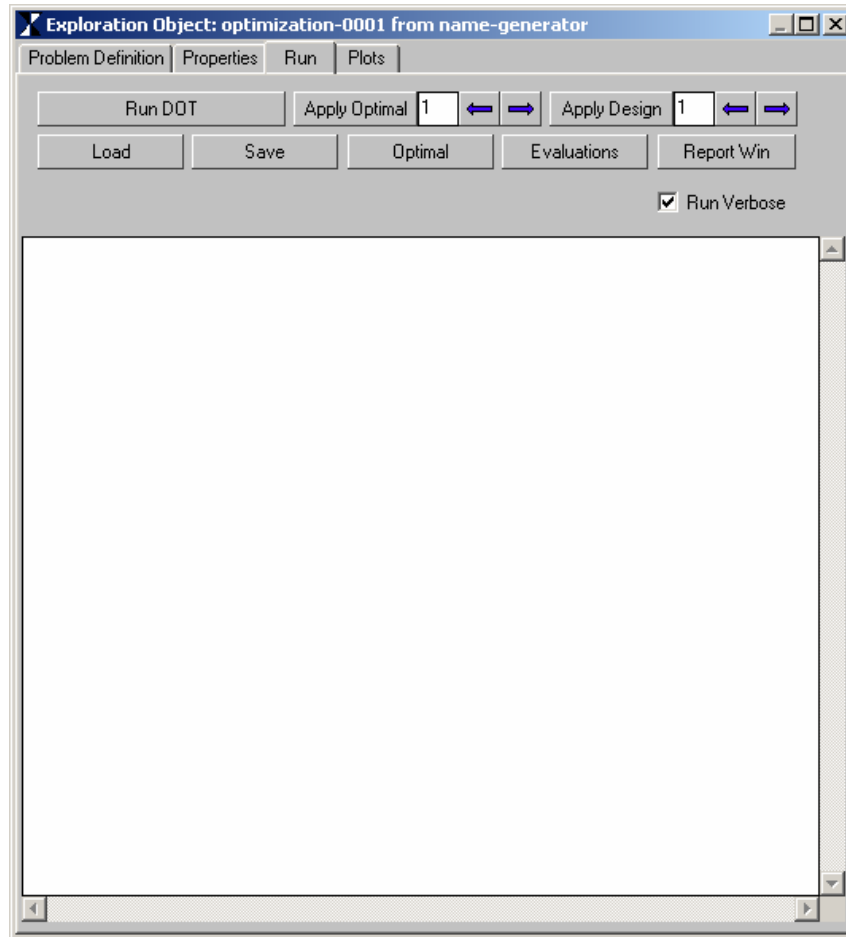


Figure 5-9: Run Tab

The designer only has to hit the “Run DOT” button, and the optimization run will begin.

Last, the user can create a report window or plots from the optimization run. These plots include “Variable vs. Objective” and “Variable/Objective vs. Iteration” plots. The “Report Window” option is utilized in the “Run” tab while the plots are completed using the “Plots” tab. Figure 5-10 illustrates the “Plots” tab.

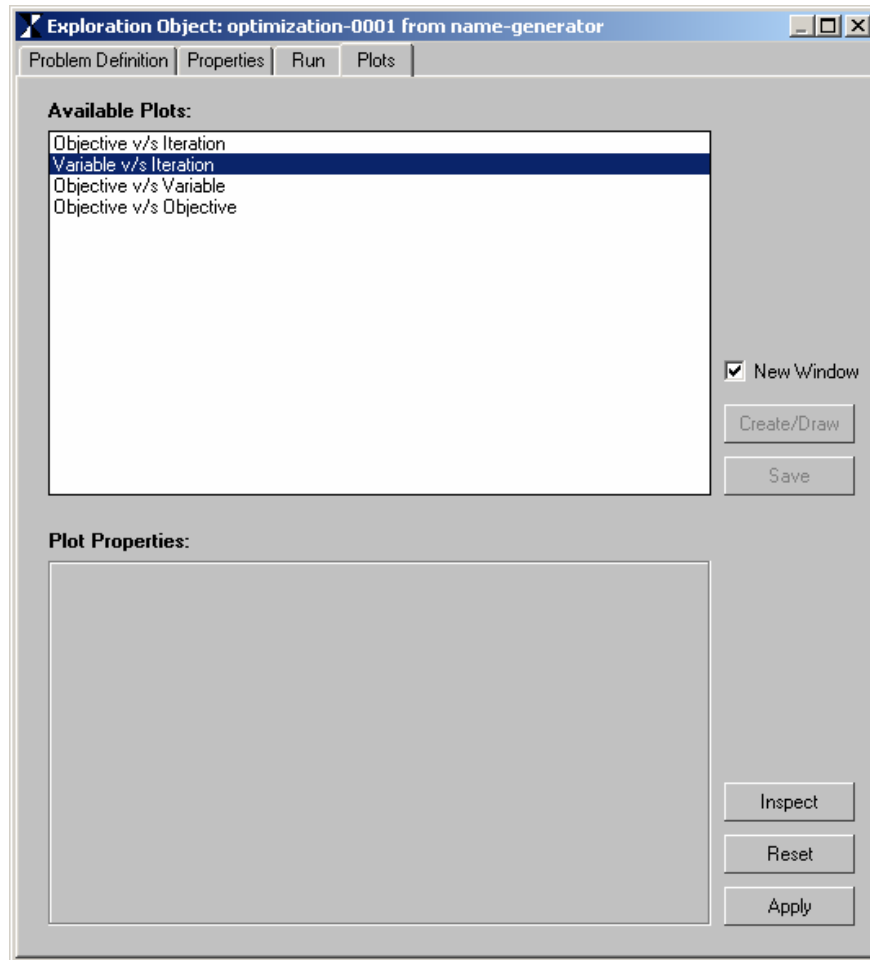


Figure 5-10: Plots Tab

5.3.1.3 Optimization Run

Once a study is setup the optimization run is completed by the AVB. Whether the objective function is a variable within the AVB or a stress calculated by NASTRAN the AVB will do the work. It will change the configuration of the modeled aircraft to the new values and recalculate the objective. The user only has to sit back and watch and does not have to write any code for the run to work or for the NASTRAN file to run properly.

5.4 Optimization Studies

The optimization tools described so far can be used to complete an optimized design of an aircraft. In this research the AVB will be utilized to illustrate it is an excellent tool for optimization. A sensitivity study, DOE study, and DOT optimization study will be completed on different aircraft.

5.4.1 Sensitivity Analysis

The AVB employs a very simple sensitivity analysis algorithm. Each variable is run at its minimum and maximum values, while all other variables are held constant at the nominal values. The results of the objective function are calculated for each run, and the user can calculate the difference in results when a variable was run at its minimum and maximum values. Whichever variable causes the greatest change in the objective function when run at its minimum and maximum values is the most sensitive property.

5.4.1.1 Setting Up Sensitivity Analysis

Sensitivity analysis is very simple to use when using AMOPT. The following is a step-by-step process in setting up a sensitivity analysis problem.

1. The desired design variables should be created and given baseline values within the model tree. Variable creation is explained in Section 5.3.1.2, “Optimization Object” in this chapter.
2. The objective/objectives should be created within the model tree
3. Add an “exploration/optimization” object to the model tree using AMOPT.
4. Select to edit the object.
5. The optimization object menu will pop up, and the analysis can be set up. First, select the analysis method to be “Sensitivity Analysis.”
6. Next, select the design variables that will be used in the optimization study
7. Then select the objective function that will be optimized.
8. Click the run tab, and hit “Run Analysis.”

5.4.1.2 Sensitivity Study Problem Statement

In this study sensitivity analysis will be completed on the BWB that has been created in this research study. The goal will be to explore which variables minimize the weight of the BWB the greatest. Trade studies will be completed on the chord length, sweep angle, and span of each section.

5.4.1.3 Sensitivity Study Wing Model

For this study a half model of the BWB model explained in the user manual attached in Section 4.2.2 in Appendix A is used. The BWB is cut along its center-line and the right wing is used for analysis. There are some differences to note between the study model and user manual model. First, one difference between the full model explained in the user manual and the model used for optimization is the leading and trailing edge of the fuselage portion of the wing are not curved the same. However, the airfoil chord lengths, wing sections, section thicknesses, and sweep angles remained the same. Furthermore, in most BWB configuration the winglet is optional, so to decrease the number of design variables used, the winglet was dropped from the study. Figure 5-11 illustrates the wing sections and sweep angles of the BWB model.

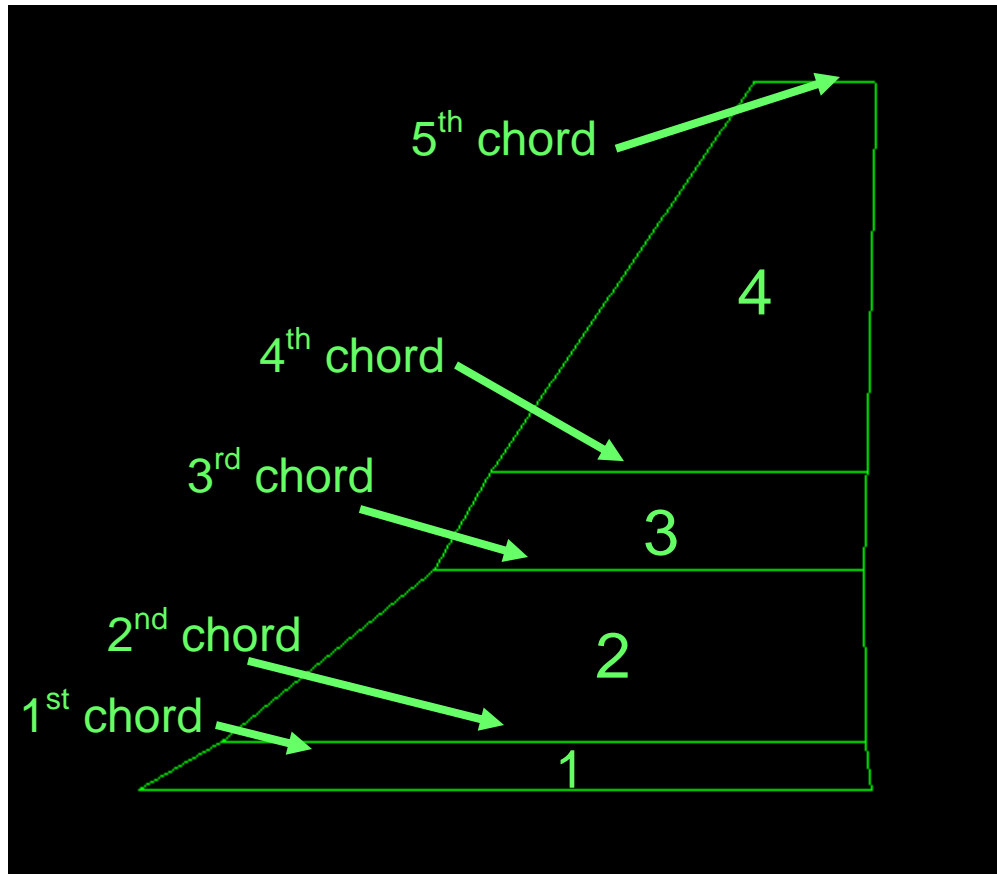


Figure 5-11: BWB Wing Planform and Section Info

5.4.1.4 Sensitivity Problem Setup

The sensitivity problem was set up as described in section 5.4.1.1, “Setting Up Sensitivity Analysis.” The weight was the objective function in this study. The weight is calculated using the “get-wing-structural-weight” command in the AVB. Once the objective function was formed the analysis was run.

5.4.1.5 Sweep Angle Sensitivity Study

The first trade study completed was changing the sweep angle of each section. Variables were created to represent the sweep angle of each section and used as design

variables in the optimization study. The weight was the objective function. Table 5-1 displays the sweep angle variables and constraints.

Table 5-1: Sweep Angle Constraints

	Minimum	Maximum
angle-one	45	75
angle-two	36	66
angle-three	15	45
angle-four	19	49

The analysis was run and the results are shown in the table AMOPT creates illustrated in Figure 5-12.

X Exploration Object: optimization-0001 (from name-generator)					
File Edit					
Exploration Evaluations					
Object : optimization-0001					
Report Date : 12-Dec-2004, 17:34.09					
	sweep-angle-one	sweep-angle-two	sweep-angle-thr	sweep-angle-fou	weight
1	45.00000	51.00000	30.00000	34.00000	87409.51201
2	75.00000	51.00000	30.00000	34.00000	89160.66530
3	60.00000	36.00000	30.00000	34.00000	87153.70641
4	60.00000	66.00000	30.00000	34.00000	90369.23047
5	60.00000	51.00000	15.00000	34.00000	84617.63340
6	60.00000	51.00000	45.00000	34.00000	87303.31883
7	60.00000	51.00000	30.00000	19.00000	87120.97204
8	60.00000	51.00000	30.00000	49.00000	83441.60228

Figure 5-12: Sweep Angle Sensitivity Table

The results were used to calculate the most sensitive sweep angle as shown in Table 5-1. The sweep angle of the fourth wing section affected the weight the greatest. It had a sensitivity weight of 3679 compared to the next closest of 3215.

Table 5-2: Sweep Angle Sensitivity Study Results

Run	angle-one	angle-two	angle-three	angle-four	weight	sensitivity
1	45	51	30	34	87409.51	
2	75	51	30	34	89160.67	1751.15
3	60	36	30	34	87153.71	
4	60	66	30	34	90369.23	3215.52
5	60	51	15	34	84617.63	
6	60	51	45	34	87303.32	2685.69
7	60	51	30	19	87120.97	
8	60	51	30	49	83441.60	3679.37

5.4.1.6 Chord Length Sensitivity Study

The chord length analysis was the second sensitivity study completed. There were five chord length variables that affect the wing. Variables were created to represent each chord length so the variables could be used in optimization. Table 5-3 displays the chord length variables and their constraints. Figure 5-13 illustrates the optimization set up.

Table 5-3: Chord Length Study Constraints

	Minimum	Maximum
Chord-one	1700	1900
Chord-two	1478	1678
Chord-three	956	1156
Chord-four	824	1024
Chord-five	200	400

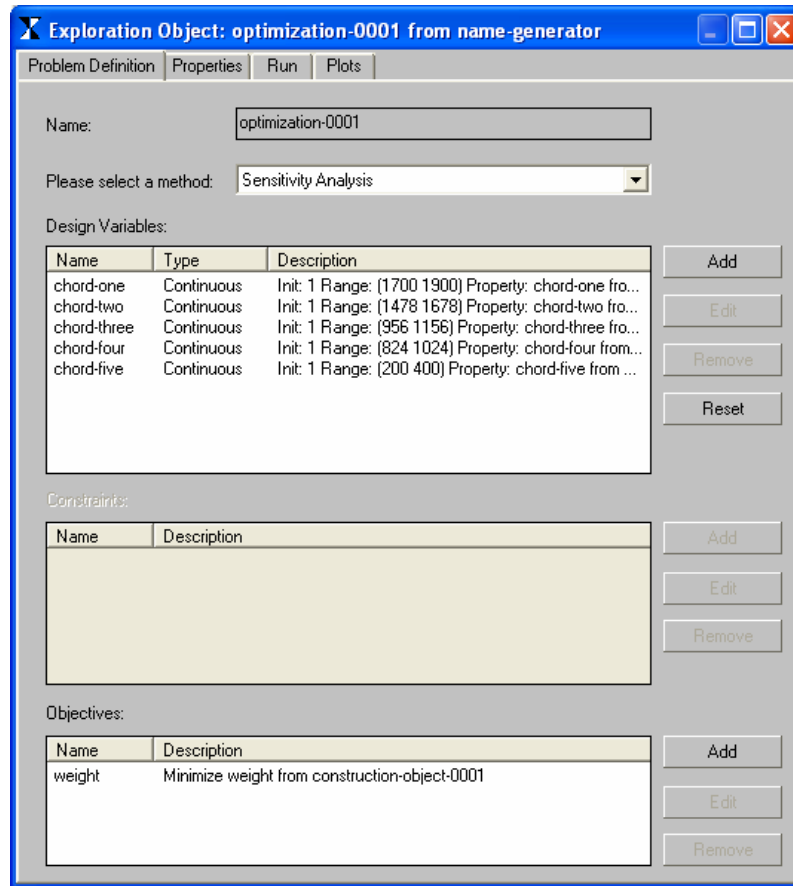


Figure 5-13: Optimization Setup

Once the set up was complete the analysis was ran. Table 5-4 contains the results of the study and highlights the most sensitive chord length. The third chord length was the most sensitive chord length, and it affects the weight the greatest. Its weight was 6171 compared to the next closest of 5300.

Table 5-4: Chord Length Sensitivity Study Results

Run	chord-one	chord-two	chord-three	chord-four	chord-five	weight	sensitivity
1	1700	1578	1056	924	300	87194.24	
2	1900	1578	1056	924	300	88501.43	1307.19
3	1800	1478	1056	924	300	85064.68	
4	1800	1678	1056	924	300	89759.11	4694.43
5	1800	1578	956	924	300	83920.18	
6	1800	1578	1156	924	300	90091.44	6171.25
7	1800	1578	1056	824	300	83977.80	
8	1800	1578	1056	1024	300	89277.44	5299.64
9	1800	1578	1056	924	200	84920.81	
10	1800	1578	1056	924	400	89816.13	4895.32

5.4.1.7 Span Sensitivity Study

The third sensitivity study completed was on the span of each wing section. Variables were created to represent the span of each section and used as design variables in the optimization study. Once the set up was complete the analysis was run. The results are displayed in Table 5-6. Table 5-5 displays the span variables and constraints.

Table 5-5: Span Constraints

	Minimum	Maximum
span-one	80	160
span-two	380	460
span-three	200	280
span-four	920	1000

Table 5-6: Span Sensitivity Study Results

Run	span-one	span-two	span-three	span-four	weight	sensitivity
1	80	420	240	960	85851.22	
2	160	420	240	960	89740.52	3889.30
3	120	380	240	960	86229.08	
4	120	460	240	960	89231.87	3002.79
5	120	420	200	960	86490.26	
6	120	420	280	960	88183.97	1693.71
7	120	420	240	920	86842.52	
8	120	420	240	1000	88057.55	1215.03

The span of the first wing section changes the weight the greatest as highlighted in table 5-6.

5.4.1.8 Sensitivity Study Conclusion

The AVB is a good tool for sensitivity analysis. The user only needs to create the aircraft model, design variables, and objective function while the AVB will set up the analysis. Once the analysis is set up, the AVB will run all the cases and give results of the objective function. The user can quickly evaluate which variable is the most sensitive variable, or in other words which variable affected the objective function the greatest. The sensitivity analysis tool is easy and quick to use, and great for optimization.

5.4.2 DOE Analysis

DOE is another optimization option available within the AVB. The user enters the design variables and objectives, and the AVB will create a full factorial list of runs to complete the DOE. The objectives will be computed as the AVB changes the variables to the proper values with in the DOE list. The user can then copy the results into Excel and create a regression plot in order to optimize the configuration. This process is made simple by the AVB because it creates the DOE runs for the user instead of the user having to enter the runs manually. Furthermore, the results are generated in a user friendly format so that the results can be pasted into Excel and regression analysis can be completed.

5.4.2.1 Setting Up DOE

The DOE analysis option is very simple to use when employing AMOPT. The following is a step-by-step process in setting up a DOE analysis problem.

1. The desired design variables should be created and given baseline values.
2. The objective/objectives should be created within the model tree
3. Add an “exploration/optimization” object to the model tree using AMOPT.
4. Select to edit the object.
5. The optimization object menu will pop up, and the analysis can be set up. First, select the analysis method to be “DOE Analysis.”
6. Next, select the design variables that will be used in the optimization study
7. Then select the objective/objectives
8. Hit the “Properties” tab to view the “DOE Run Matrix.” The user can customize or keep it.
9. Click the run tab, and hit “Run Design of Experiments.”
10. The DOE will run and generate a results table. Hit the “Evaluations” tab to see the results
11. Finally, the user can use the results to run regression analysis in Excel or whatever they desire.

5.4.2.2 DOE Problem Statement

In this study a basic trapezoidal wing will be created and optimized. The goal will be to minimize the weight and stay within Von mises stress constraints. The wing will be run thru an aerodynamic load case and the maximum Von mises stress and weight will be calculated. The results will be taken to Excel and regression analysis will be completed and the model optimized. This study will be used to demonstrate the capability of the AVB to employ DOE to optimize an aircraft configuration.

5.4.2.3 DOE Wing Model

A basic trapezoidal wing has been used to assess many aerodynamic equations. This study employs the trapezoidal wing whose planform is shown in Figure 5-14 to complete a DOE analysis. The nominal dimensions are given in Table 5-7. The nominal weight is 1371 pounds and the nominal Von mises stress was 7105 pounds per square inch (psi).

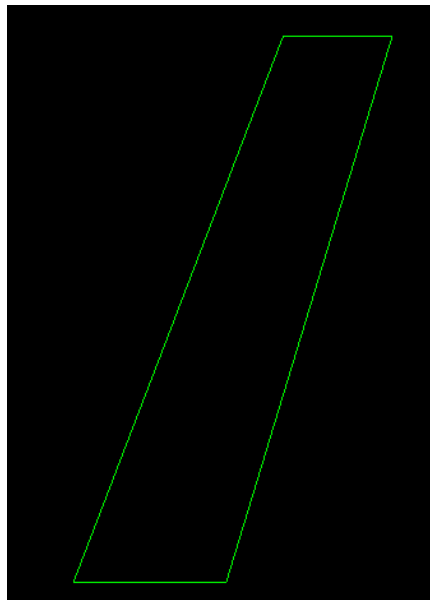


Figure 5-14: Trapezoid Wing Planform

Table 5-7: Trapezoidal Wing Dimensions

Variable	Value	Units
Root-chord	84	inches
Tip-chord	60	inches
Sweep-Angle	20	degrees
Wing-Span	300	inches

5.4.2.4 DOE Wing Design Variables

The root chord, tip chord, sweep angle, and wing span will be the design variables used to optimize the weight in this study.

5.4.2.5 DOE Wing Design Constraints

The Von mises stress of each element will be constrained to be less than 10000 pounds per square inch (psi) in this study. Table 5-8 shows the design variable constraints.

Table 5-8: DOE Design Constraints

Variable	Min	Max	Nominal	Units
Root-chord	36	120	84	inches
Tip-chord	24	84	60	inches
Sweep-Angle	0	45	20	degrees
Wing-Span	180	480	300	inches

5.4.2.6 DOE Wing Objective

The goal of this study will be to minimize the weight of the wing.

5.4.2.7 DOE Problem Setup

The DOE was set up using the instructions given in section 5.4.2.1, “Setting Up DOE.” The weight and Von mises stress were used as the objective functions in this analysis. The Von mises stress is calculated by running NASTRAN analysis on the model under the given load conditions. The stresses are calculated and formatting into a table within the AVB called “stresses-mesh-with-color-mapping-table.” The user points to this table and calculates the maximum stress. Once the weight and Von mises objective functions are formulated, the full factorial run matrix is setup by the AVB and the analysis is run.

5.4.2.8 DOE Run Matrix

Figure 5-15 shows the DOE run matrix used for the full factorial study on 4 design variables. This run matrix was created by the AVB and the user did not have to enter any values saving engineering time.

Customize DOE Run Matrix

Add Remove Default

	root-chord	tip-chord	sweepangle	wing-span
1	36	24	0	180
2	36	24	0	480
3	36	24	45	180
4	36	24	45	480
5	36	84	0	180
6	36	84	0	480
7	36	84	45	180
8	36	84	45	480
9	120	24	0	180
10	120	24	0	480
11	120	24	45	180
12	120	24	45	480
13	120	84	0	180
14	120	84	0	480
15	120	84	45	180
16	120	84	45	480
17	78.0	54.0	22.5	330.0

Figure 5-15: DOE Run Matrix

5.4.2.9 DOE Results

Once the DOE Run Matrix was generated the DOE was run. The results are shown in Table 5-9.

Table 5-9: DOE Results

Run	root-chord (in)	tip-chord (in)	sweep-angle (degrees)	wing-span (in)	Weight (lbs)	Vonmises (psi)
1	36	24	0	180	605.69	11584.16
2	36	24	0	480	1515.54	26367.38
3	36	24	45	180	783.74	15460.21
4	36	24	45	480	1999.03	31509.55
5	36	84	0	180	792.07	6440.74
6	36	84	0	480	1900.47	17895.84
7	36	84	45	180	1000.66	11943.80
8	36	84	45	480	2426.51	24000.87
9	120	24	0	180	886.61	7685.40
10	120	24	0	480	2079.93	11450.35
11	120	24	45	180	1034.38	8166.08
12	120	24	45	480	2557.06	14970.38
13	120	84	0	180	1063.09	4309.37
14	120	84	0	480	2462.09	6729.88
15	120	84	45	180	1248.32	4416.14
16	120	84	45	480	2984.60	9000.52
17	78	54	22.5	330	1472.48	8482.77

5.4.2.9 Regression Analysis of DOE Results

A nice point about the AVB is that its DOE evaluation results are outputted in table format. This allows the designer to quickly copy and paste the results into Excel to complete regression analysis and then use solver to optimize the model.

For this study the DOE results were inputted into Excel and the weight and Von mises stress outputs were estimated using regression analysis. The root-chord, tip-chord, and wing-span variables shown were used as the x-variables (independent variables) for

the regression analysis, and the weight and Von mises stress variables were used as the y-variables (dependent variables). The regression analysis coefficient results are shown in Table 5-10.

Table 5-10: Regression Analysis Coefficients

	Weight Analysis	Von Mises Analysis
<i>Design Variable</i>	<i>Coefficient</i>	<i>Coefficient</i>
y-intercept	-692.5103166	15273.85296
Root-chord	4.899359003	-116.7774324
Tip-chord	5.033011062	-88.45072021
Sweep-angle	7.58002925	75.01231006
Wing-span	4.379453088	29.96619751

5.4.2.10 Using The Solver

Once regression analysis is run the designer can use the variable coefficients to optimize the design. The Solver within Excel is very quick at optimizing the design. In this study the regression coefficients were used and the design optimized using the Excel Solver. The same design constraints displayed in Table 5-8 were used in Excel along with making sure the Von mises stress was less than 10000 psi. The optimization results are noted in Table 5-11 and the optimized design displayed in Figure 5-16. The optimized design can be compared to the baseline model shown in Figure 5-14.

Table 5-11: DOE Solver Results

Variable	Baseline values	Optimized values
<i>root-chord</i>	84 inches	73.17 inches
<i>tip-chord</i>	60 inches	24 inches
<i>sweep-angle</i>	20 degrees	0 degrees
<i>wing-span</i>	300 inches	180 inches

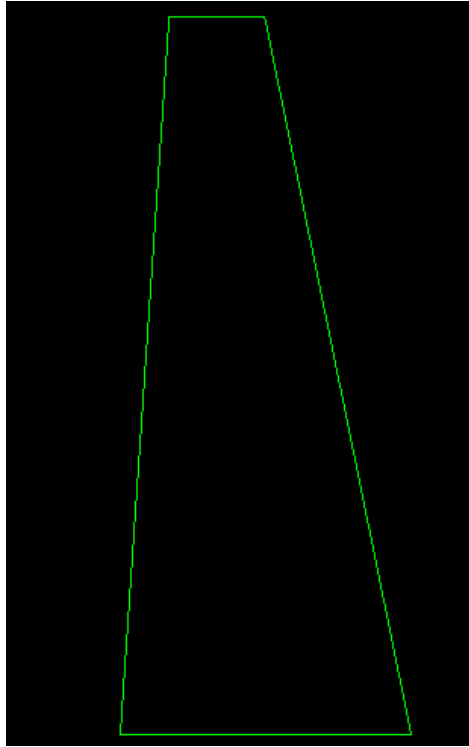


Figure 5-16: DOE Optimized Model

The DOE analysis using the solver is an estimated optimization and not exact. However, the optimized variable values can be entered back into the AVB, and analysis rerun to check the true results.

5.4.2.11 DOE Conclusion

The DOE tool within AVB is very useful in optimization. Once the design variables, constraints, and objectives are made the AVB will do the rest of the work. The user can select what type of DOE run matrix should be produced: 2nd order full factorial or 3rd order full factorial. Once the method is chosen, the AVB creates the run list for the user, and the user only needs to run the analysis. The AVB will run the full matrix and produce the results in table format. This is constructive because it allows quick integration with external programs that require results to be tabular. For example, DOE run results could be pasted into Microsoft Excel and regression analysis used to complete

the optimization. The user would not have to enter all the results in by hand, but could just quick copy and paste the results from the AVB into Excel. Once the results are in Excel and regression is run, the user can quickly optimize the design using the Excel Solver. This is useful because it allows for a quick optimization method, rather than having to wait for a full optimization algorithm to run. Thus, the AVB is a good tool for DOE optimization.

5.4.3 DOT Optimization

The AVB employs the DOT optimization tool as one of its optimizers. The DOT tool uses the feasible directions algorithm to find the optimized configuration. Without an extra license, the AVB only allows three design variables to be used with the DOT tool.

5.4.3.1 Setting Up DOT Optimization

The DOT optimization algorithm is simple to use when employing AMOPT. The following is a step-by-step process in setting up a DOT analysis problem.

1. The desired design variables should be created and given baseline values within the model tree.
2. The objective/objectives should be created within the model tree
3. Add an “exploration/optimization” object to the model tree using AMOPT.
4. Select to edit the object.
5. The optimization object menu will pop up, and the analysis can be set up. First, select the analysis method to be “DOT”
6. Next, select the design variables that will be used in the optimization study
7. Then set up the design constraints
8. Next, select the objective function that will be optimized.
9. Now select the “Properties” tab and edit the optimization design.
10. The user can change the max number of iterations, step size, extreme value, etc., as shown in Figure 5-17.

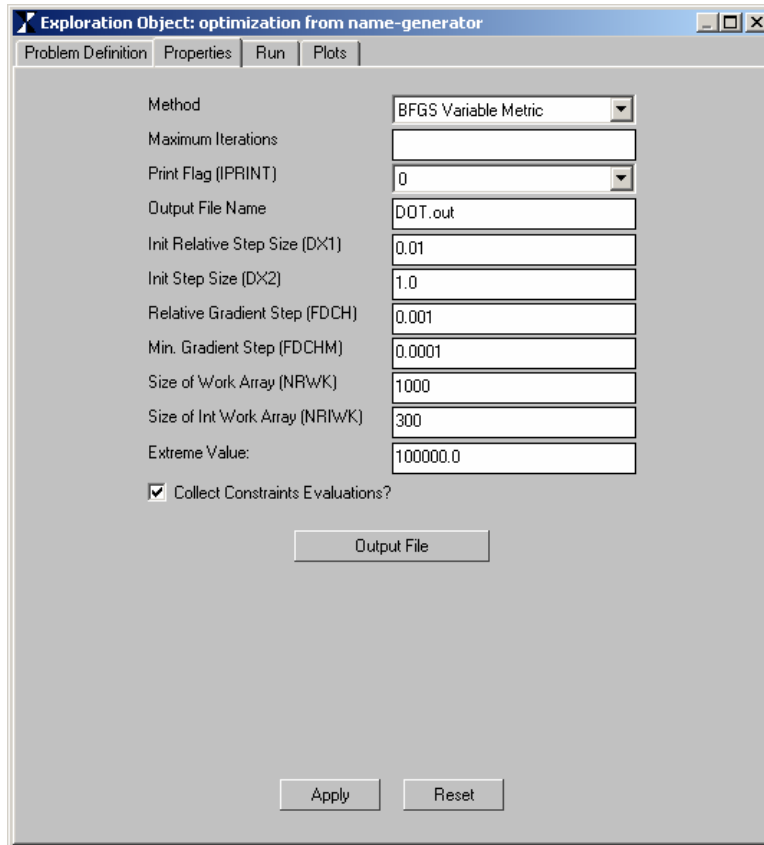


Figure 5-17: DOT Properties Tab

11. Once the DOT set up is completed, click the “Run” tab, and hit “Run DOT.”

5.4.3.2 DOT Optimization Problem Statement

In this optimization study the DOT optimization tool will be used to optimize the trapezoidal wing described in the DOE wing study. Without an extra license the AVB allows only three variables to be optimized. The root chord, tip chord, and wing span will be chosen as design variables and used to optimize the weight, while the Von mises stress will be constrained.

5.4.3.3 DOT Optimization Wing Model

The same wing model described above in section 5.4.2.3, “DOE Wing Model” is used for this study except the sweep angle of the wing remained constant at 20 degrees.

5.4.3.4 DOT Optimization Problem Set Up

The DOT optimization problem was set up as described in section 5.4.3.1, “Setting Up DOT Optimization.” The nominal design variable values are given in Table 5-12. The nominal weight was 1371 pounds and nominal Von mises stress was 7105 psi. A limit of 25 iterations was given with a step size of 20. The 25 iterations were chosen because this study is being used to illustrate the capability of the AVB to us DOT optimization rather than fully optimize the model.

Table 5-12: DOT Optimization Design Variables

Variable	Nominal	Units
Root-chord	84	inches
Tip-chord	60	inches
Wing-Span	300	inches

5.4.3.5 DOT Optimization Design Variables

The root chord, tip chord, and wing span will be used as design variables in this study.

5.4.3.6 DOT Optimization Design Constraints

The Von mises stress of each element will be constrained to be less than 10000 psi in this study. Table 5-13 shows the design variable constraints.

Table 5-13: DOT Optimization Design Constraints

Variable	Min	Max	Nominal	Units
Root-chord	36	120	84	inches
Tip-chord	24	84	60	inches
Wing-Span	180	480	300	inches

5.4.3.7 DOT Optimization Results

Table 5-14 displays the results of the DOT Optimization run matrix. The optimal run out of the 25 is highlighted. The root chord was optimized to be 46.5 inches, the tip chord was optimized to be 37.1 inches, and the wing span was optimized to be 230.5 inches. The weight was minimized to 880 pounds, and the Von mises stress was 9999 psi. Figure 5-18 shows the new wing planform shape after the DOT optimization run. The new planform can be compared to the baseline planform depicted in Figure 5-14. The new model is more slender and shorter.

Table 5-14: DOT Optimization Run Matrix

Run	root-chord (inches)	tip-chord (inches)	wing-span (inches)	weight (lbs)	vonmises (psi)
1	84	60	300	1391.74782	7105.5708
2	84.084	60	300	1392.12552	7101.87695
3	84	60.06	300	1392.01121	7101.99707
4	84	60	300.3	1392.99454	7111.44922
5	72.52298	54.28326	284.96667	1254.60843	7708.02002
6	53.95277	45.03338	260.64223	1046.20909	9257.09961
7	36	24	196.96001	685.67746	11759.67969
8	45.82495	40.98489	249.9959	960.2501	10417.46973
9	54.00672	45.03338	260.64223	1046.42099	9249.72266
10	53.95277	45.07841	260.64223	1046.39665	9254.07715
11	53.95277	45.03338	260.90287	1047.15654	9264.43066
12	36	35.68586	237.10903	859.00311	12385.79004
13	49.68992	42.81383	255.05432	1000.30819	9851.58789
14	48.71845	42.30801	253.78087	989.96864	9993.23926
15	48.67263	42.28415	253.72081	989.48239	9999.99902
16	48.7213	42.28415	253.72081	989.66931	9992.97656
17	48.67263	42.32644	253.72081	989.6579	9997.125
18	48.67263	42.28415	253.97453	990.38131	10007.48047
19	46.38636	37.09866	230.58122	879.56314	10020.62012
20	46.50826	37.12796	230.54591	879.98762	9997.86426
21	46.49563	37.12493	230.54956	879.94366	10000.0498
22	46.49593	37.125	230.54948	879.94471	10000.0498
23	46.49622	37.12507	230.54939	879.94571	9999.92188
24	42.6871	28.70837	193.14057	710.17964	10407.41016
25	45.09558	29.28726	192.44296	717.34406	10223.7002

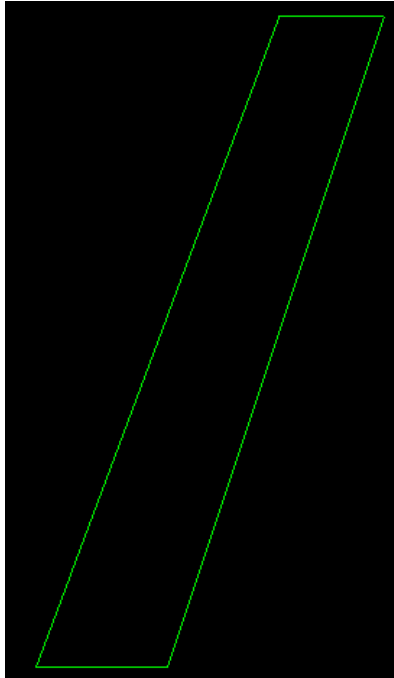


Figure 5-18: Optimized Trapezoid Wing Planform

5.4.3.8 DOT Optimization Conclusion

The DOT optimization tool is simple to use when employing the AVB. The user only needs to enter the design variables, constraints, and objective function and then run the analysis. The AVB will use the feasible search direction method to find the optimal solution.

5.5 Model Validation

A model check was needed to evaluate if the AVB is modifying the model correctly as it changes configurations during optimization runs. The main area of concern was if the spar and rib configuration was remaining the same throughout the configuration changes. Figure 5-19 illustrates the model used for this verification.

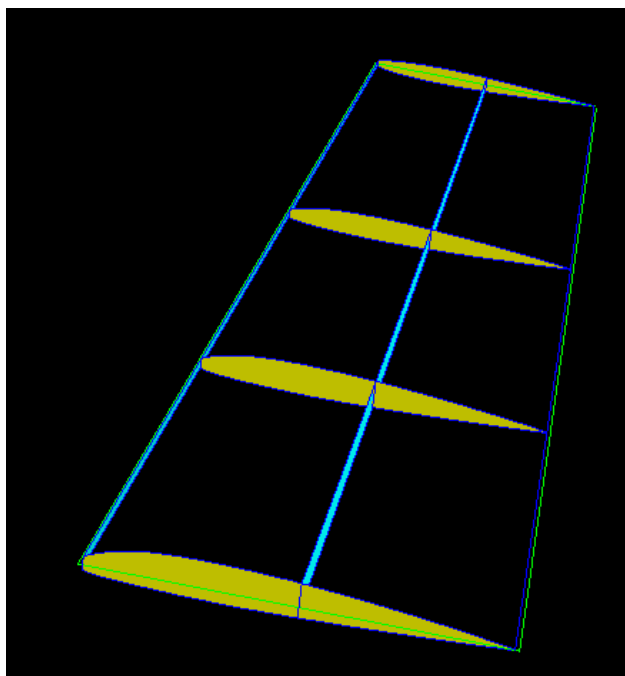


Figure 5-19: Comparison Model

5.5.1 Configuration Change Check

The trapezoidal wing shown in Figure 5-19 was used above and its configuration changed as shown in Figure 5-20. The root chord, tip chord, and sweep angle were all altered and the ribs and spars remained in the same configuration they were before the change. The AVB changes the model correctly without affecting the spar and rib configuration, proving that the user can trust the AVB during optimization run model changes.

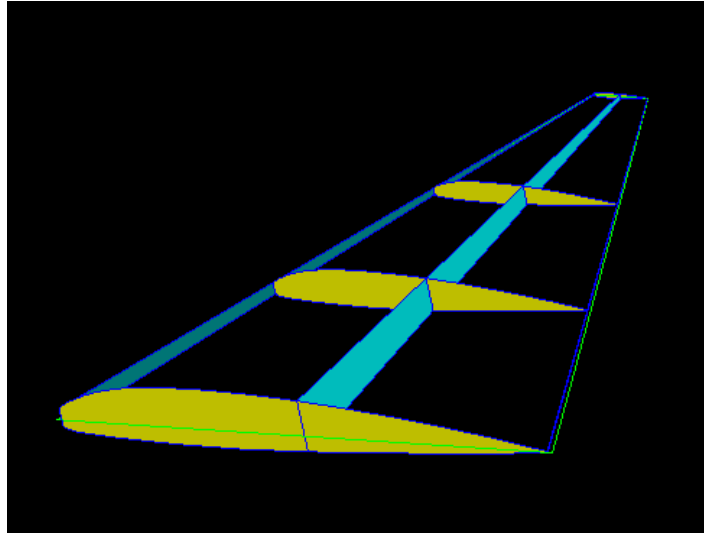


Figure 5-20: Altered Model

5.6 Optimization Conclusion

The AVB is a great tool for design optimization giving many different options to the user. First, the AVB can create and run an optimization NASTRAN input file of the model create by the engineer so that the engineer does not have to enter any input cards. This saves time and energy for the designer. Second, many different optimization tools are available in the AMOPT toolbox. Sensitivity analysis can be completed so the user can find the most sensitive design variables for the model and reduce the number of variables. Plus, a DOE option is available to run and easily integrate with Excel so a quick optimization of the aircraft can be completed rather than having to wait on a optimization algorithm to be completed. Finally, the AMOPT toolbox contains the DOT optimization method which employs the feasible direction algorithm to optimize a design. All of these optimization options are easily used and the majority of the work is done by the AVB minimizing the time and energy of the user. The AVB is an awesome optimization tool.

6. SUMMARY

6.1 Summary

The United States Air Force set up a Simulation-Based Research and Development (SBRD) team in an effort that seeks to improve the assessment method for new technologies by enabling improved connectivity between people and analysis tools. It seeks to implement commercial-off-the-shelf software whenever possible, with hopes of integrating various programs. The SBRD team should create an application that has a small learning curve so that users of all backgrounds will find it useful. The AVB is a design tool that meets these requirements. The AVB allows quick baseline modeling of an airplane, and can be employed to tie analysis codes together to complete a design of an aircraft, including optimization. The AVB has a small learning curve so that beginning aircraft designers will find it easy to use. Furthermore, the AVB environment aids in making the modeling process quicker since it contains available objects to build an aircraft. It is recommended that further research be completed on the AVB so that it can be employed for use by the United States Air Force.

7. FUTURE WORK

7.1 Future Work

There are many items that could be explored with the AVB for future work. First, the meshing capabilities could be explored to investigate if desired meshes can be created by the engineers. In other words, the AVB should be evaluated to see if local meshes can be made more fine, while global meshes can be created less fine. Second, it would be good to explore the ability of the AVB to be integrated with the internet. For example, one might be interested in storing AVB created files on the net so a tool to upload the files should be created. Furthermore, one might be interested in writing variables and their values to an XML file to store on the net, so code for this should be written. Lastly, objects for landing gear, missiles, and engines could be formed and utilized in the AVB to further its integration capabilities.

8. BIBLIOGRAPHY

1. Ko, Yan-Yee Andy, "The Multidisciplinary Design Optimization of a Distributed Propulsion Blended-Wing-Body Aircraft", Phd. Dissertation, Apr. 2003.
2. Veley, Duane E., Blair, Maxwell, Zwebert, Jeffrey V., "Aerospace Technology Assessment System", AIAA 98-4825, 1998.
3. Wakayama, Sean, "Multidisciplinary Design Optimization Of The Blended-Wing-Body", AIAA 98-4938, Sept. 1998.
4. Stevenson, Mark D., "Multidisciplinary Conceptual Vehicle Modeling", Thesis Dissertation, May 2001.
5. Adaptive Modeling Language Reference Manual: Version 3.3.2, Technosoft Inc., Cincinnati, OH, 2002.
6. Blair, M., "Enabling Conceptual Design in a Technology-Driven Environment", AIAA paper 98-4741, presented at the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, Sept. 2-4, 1998.
7. TechnoSoft Inc. web site, <http://www.technosoft.com>
8. Mizrahi, Joe. "Flight To The Future." Wings April 1999, Vol. 29, No. 2, Sentry Magazines, Granada Hills, CA, pp. 8-19.
9. Wakayama, S., "Blended-Wing-Body Optimization Problem Setup," AIAA Paper 2000-4740 4003 presented at 8th AIAA Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, September 6-8, 2000.
10. Chemaly, A., and Marino, J. "Structural Design, Analysis, Optimization, and Cost Modeling Using The Adaptive Modeling Language", AIAA 02-1296, 2002.
11. Veley, Duane E., "Optimization in the Adaptive Modeling Language", AIAA, 98-4872, 1998.
12. Nicolai, L.M., Fundamentals of Aircraft Design, METS, Inc., San Jose, CA, 1984

