COEUS: Concurrent Engineering Utility for Systems

A project present to
The Faculty of the Department of Aerospace Engineering
San Jose State University

in partial fulfillment of the requirements for the degree
Master of Science in Aerospace Engineering

By

Ian C. Dupzyk

December 2012

approved by

Dr. Periklis Papadopoulos
Faculty Advisor

San José State
UNIVERSITY

The Undersigned Faculty Committee Approves

COEUS: Concurrent Engineering Utility for Systems

by

Ian C. Dupzyk

APPROVED FOR THE DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING

12/12/12

Dr. Periklis Papadopoulos, San Jose State University (Committee Chair) Date

12/13/12

Marcus Murbach, NASA Ames Research Center (Advisor) Date

14 Dec 12

Dr. Nikos Mourtos, San Jose State University (Advisor) Date

Abstract

To meet the demands of engineering complex systems it is becoming more and more important to incorporate Concurrent Engineering into the design of complex systems. This report outlines the work performed to develop a multidisciplinary system design tool named COEUS. The purpose of COEUS is to help facilitate Concurrent Engi-neering in the early design phases of complex systems. This goal is accomplished by providing a single environment where each discipline and subsystem can be owned by a different group, but executed in an integrated fashion. This allows ownership of a given subsystem or module to go to the subject matter experts while eliminating the delay in communication and design iterations. To demonstrate the system developed herein, an analysis model was created to analyze a rigid airship. This model incor-porates a range of fidelities in the analysis modules. Some modules are low fidelity that use empirical relations to determine subsystem masses or component costs. One module uses direct simulation to determine structural mass through direct simulation with a finite element tool. The use of this tool allows a user to perform analyses, sensitivity studies, and trade studies to gain greater understanding of a system and how it is influenced by different variables.

Contents

List of Figures

List of Tables

Nomenclature

$_{comb}$ Combustion Efficiency

$_{ins}$ Insulation material density

$_{out}$ External environment density

$_{skin}$ Areal mass of the skin material

C_d Drag coefficient

E Endurance

F_{drag} Vehicle drag force

f mf Fin mass fraction used to compute the fin mass as a fraction of the structure mass

H_c Buoyancy Fuel Heat of Combustion

h_{loss} Heat flux loss through the vehicle skin

K_{ins} Insulation Thermal Conductivity

m_{bFuel} Mass of buoyancy fuel

m_{bTank} Mass of buoyancy fuel storage tank

m_{fin}	Mass of the fin assembly
m_{ins}	Insulation mass
M_{mass}	Mass Margin
m_{pFuel}	Mass of propulsive fuel
m_{skin}	Vehicle skin mass
m_{str}	Structure mass
M_{time}	Time margin
N_{beams}	Number of beams in the system
N_{sup}	Number of beam supports in the system
P_{prop}	Propulsive power
q_1	Free stream dynamic pressure, defined as 0:5 v_{12}
S_{drag}	Drag reference area
S_{skin}	Area
$T_{beamFab}$	Time to fabricate all the structural beams of the system
T_{end}	Time to join a single end of a beam
t_{ins}	Insulation Thickness
T_{in}	Temperature Inside the Envelope
T_{joint}	Time to fasten a single end of a support
T_{out}	Temperature Outside the Envelope

tmf	Buoyancy fuel tank mass fraction
v_1	Free stream velocity
BSFC	Brake Specific Fuel Consumption
CAD	Computer Aided Design
CE	Concurrent Engineering
COEUS	COncurrent Engineering Utility for Systems
DSM	Design Structure Matrix
FE	Finite Element
GUI	Graphical User Interface
ICE	Integrated Concurrent Engineering
MER	Mass Estimating Relation
OML	Outer Mould Line
STP	Standard Temperature and Pressure
UI	User Interface
UOM	Unit of Measure

Chapter 1

Introduction

1.1 Motivation

Throughout history humans have created increasingly complex things from the great pyramids to automobiles to modern computers. In many cases, new approaches and procedures are needed to successfully complete such projects. For example, completion of projects like the great pyramids marked early emergence of sophisticated management and organization [6]. Automobiles marked the development of mass production enabling many people to own something that was once reserved only for the wealthy [8]. Finally, modern computers and electronic devices have illustrated the need for mass customization and the ability to create complex products in a fluid, fast-paced environment. These examples are to illustrate the general trend that products are getting increasingly more complex and the need for expedited design modifications and improvement is increasing. To address this change in environment many companies are changing their approach to design and manufacturing by adopt-ing concurrent engineering and lean practices [1]. The project described herein is the development of a tool who's purpose is to maximize the capability of concurrent engineering in an analysis driven environment and demonstrate what benefits can be attained.

1.2 Objectives

The objectives of this project are two fold; to develop an integrated analysis archi-tecture that is conducive to any level of analysis fidelity and to apply this tool to an example of moderate fidelity sizing and analysis tool for a rigid airship.

1.3 Literature Search

There are a number of multidisciplinary systems analysis tool frameworks in existence, both commercial and open source. An example of each is Phoenix Model Center and Sandia's Dakota. Dakota (Design Analysis Kit for Optimization and Terascale Applications) [5] is a tool set that is designed to provide an extensible and flexible interface for analysis programs and iteration methods. ModelCenter by Phoenix Integration [4] is a similar tool set with less capability but provides a graphical user interface.

Chapter 2

Concurrent Engineering

Concurrent Engineering (CE) is the process of performing tasks simultaneously or with a large degree of schedule overlap that would normally be performed sequentially. In the manufacturing world, CE is defined as simultaneously designing the product as well as the process and tools required to manufacture the product [1]. In a more broad sense concurrent engineering simply means to address all aspects of the products life cycle from the project's onset.

In the implementation of concurrent engineering all aspects of the project are in a state of flux. This is especially valuable when many components are dependent on one another creating a highly interdependent system. The primary benefits of adopting CE in a complex project is savings in time (schedule) and cost. By accounting for all aspects of a project from the beginning one greatly reduces the potential for rework. Also, by not waiting for one phase to complete before starting the next can greatly impact and reduce schedule. This also allows design iterations to occur on component or subsystem level and not at the project phase level. With a traditional waterfall approach, the development may reach the end before an iteration need is realized. This iteration need may be because the design doesn't perform as intended or the needs for the system have changed. Therefore, multiple iterations may be required to converge the system to a satisfactory level. This would likely result in a slip in schedule or a poor quality product if a schedule slip is not viable. Since all aspects of a project are addressed simultaneously in a concurrent engineering environment, when a problem or conflict arises the iteration performed is at a component level of the system, not the entire system. This allows for both iterations of shorter duration and requiring fewer resources to complete.

CE is also referred to as Integrated Concurrent Engineering (ICE) and can be defined as the process of engineering a product or system with a high degree of parallel tasks. In contrast, the traditional approach is to perform interdependent tasks in series, commonly referred to as the Waterfall method for development. Waterfall is generally used for development projects where the requirements and the technologies involved are understood well. For other projects that are not so straight forward this method can be problematic as repeating a previous step becomes difficult and costly [7].

It is a common practice to execute non-dependent tasks in parallel to save time; however, as the degree of parallelism increases for dependent tasks, there are several problems that may arise. The primary objective of CE is to facilitate faster iterations in the development of a particular product. In terms of time to market (or project completion) and cost, the Waterfall development strategy would be the least expensive. However, the waterfall approach is unrealistic in that highly complex projects never flow in a single direction void of iteration cycles. When iterations are considered, the waterfall method can take substantially longer than other methods and no longer be the least in cost. In this real scenario, a project with a high degree of concurrent tasks would likely be shorter in schedule time to completion and also lower in total cost. Since 1980, Concurrent Engineering has demonstrated this fact of allowing products to be developed and reach market faster and at lower cost that traditional approaches.

Chapter 3

COEUS

3.1 Scope of Work

The scope of work undertaken in this Master's project was to create a tool that would better facilitate Concurrent Engineering in product and system design. This tool would provide capability of analyzing entire systems in an integrated approach to allow for greater understanding of system characteristics and the sensitivities of the system to different variables. Specifically, the types of analysis that will be de-signed into COEUS will be simple analyses, sensitivity studies, trade studies, and optimization. Currently, there are other tools that are designed for integration of multidisciplinary system analysis such as ModelCenter and Dakota. These tools are broad in their applications whereas the tool developed in this project was intended to be more specific to the multidisciplinary design of engineering systems. This architecture would differ from these existing architecture in that these would be designed with the needs of systems engineering projects in mind. Meaning that the target project would be one in which the majority of people setting up and using the tool are more physical engineers as opposed to computer engineers. Furthermore, there are specific types of analysis that would be commonly performed, such as trade, sensi-tivity, optimization, and Monte Carlo studies. The current uses of these architectures are used more in the conceptual design of complex systems. This project intends to not only match this capability but demonstrate the capability to evolve in com-plexity, facilitating use of this integrated environment beyond the conceptual stage of development.

3.2 Program Name

The program developed herein was intended to facilitate concurrent engineering from early conceptual stages of development through detailed designs. Its primary purpose is to facilitate rapid loop communication by all subsystems. This is achieved through analysis being integrated allowing analysis and incorporation of results at one time. The program was named after Coeus, the Greek titan of intelligence. COEUS, stands for COncurrent Engineering Utility for Systems. COEUS facilitates concurrent engi-neering in the engineering of systems by managing communication between different analysis modules and executing the different modules in an order defined by user input.

3.3 Design Environment

At this stage in the development, COEUS is designed to be executed as a command line tool. This was chosen for a number of reasons. First, at this early stage it is easier to develop a program for command line execution. Second, some of the analysis that one may wish to perform with this tool may require multi-processor applications. As such, COEUS should be capable of running on a cluster or supercomputer. Many of which are primarily accessible via command line. Finally, a program executed by command line allows a user interface (UI) to be written in any language for any system. The UI can be written in Java, QT4, or even by a web interface depending on what the needs of the code is later in its development.

3.4 COEUS Analysis Architecture

Regardless of the type of analysis a user is performing; whether it be a simple analysis, sensitivity study, trade study, or other. The way the model is analyzed is the same, i.e. all the modules in the analysis need to be analyzed from start to finish and coupled modules will need to be iterated upon. The difference is in what variables are traded and what their different values are.

One of the key advantages of COEUS is the way it performs different types of analysis. The actual analysis is performed by a function within the program called analyze. A flow chart depicting the logic and process of the analyze function is shown in Figure 3.1. This approach allows the same analysis to be performed multiple times within a single COEUS execution with repeatable results. Furthermore, this allows the type of analysis to be controlled from the architecture and not the model. A flow diagram for the main function representative of the analysis, sensitivity, and trade study analysis types in COEUS is shown in Figure 3.2. In essence, this means that when a user sets up an analysis model, there are no modifications that need to be made to that model in order to run a sensitivity study rather than a simple analysis. With the aforementioned tools currently available, this is not the case. This approach maintains the integrity of the analysis by eliminating the chances of unintended changes due to modifying an analysis model for a different type of analysis or creating separate models for the different analyses.

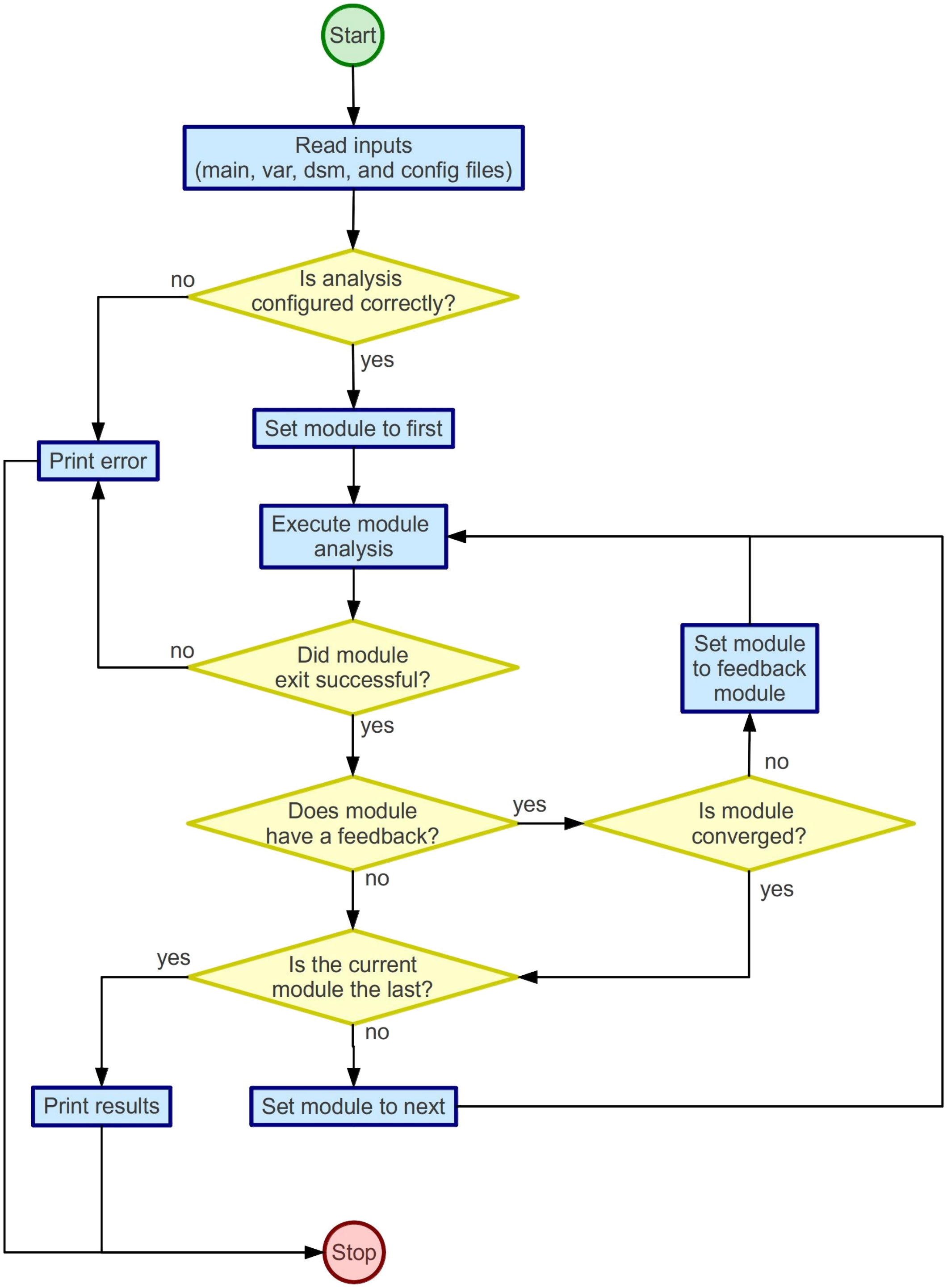

Start
Read inputs
(main, var, dsm, and config files)
Is analysis configured correctly?
no
yes
Print error
Set module to first
Execute module analysis
Did module exit successful?
no
yes
Set module to feedback module
no
Does module have a feedback?
yes
Is module converged?
no
yes
Is the current module the last?
yes
no
Print results
Set module to next
Stop

Figure 3.1: Flow diagram of the analysis function in COEUS

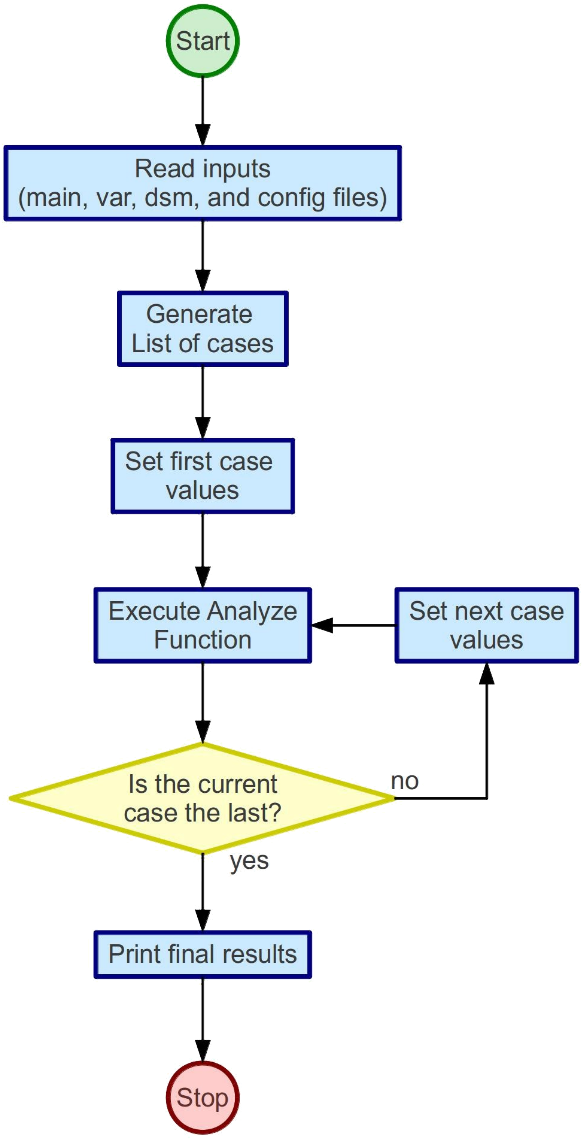


Figure 3.2: Flow diagram of the main function in COEUS

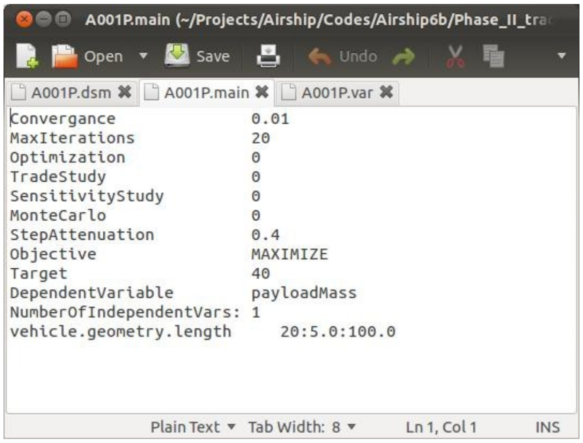
3.5 COEUS Control Files

COEUS is controlled by a number of simple text files. As with anything, there are advantages and disadvantages to having multiple files to control the program execution as opposed to a single monolithic file. The reason a single file was not chosen was for flexibility and simplicity. Having multiple files may seem complicated; however, the benefit is that the format and ease of editing each file is improved by not having to deal with complicated file formatting and sectioning. The files used here are plain ASCII text files. As this early stage version of the program is designed to be setup and run via command line, reducing or eliminating errors due to formatting problems in the input file seemed to surpass the need for a limited number of files. Furthermore, having a separate control file for each module is valuable for being able to quickly include established modules into an analysis. It may be appropriate at a later time to use an XML input format if COEUS goes to a Graphical UI.

There are two types of files that COEUS requires. The first are system level files that control the analysis, execution, and variables. The second type is the module configuration files. The latter controls what information from the system is com-municated to and from the modules. The system file naming convention is CASE-NAME.ext where \CASENAME" is chosen by the user to represent the project being analyzed and \ext" is one of main, dsm, or var to denote the main, design struc-ture matrix or n-square diagram, or variable declaration files, respectively. For the module configuration files the naming convention is NAME.config, where \NAME" is the module name (e.g. STRUCTURES.config) without spaces or special_characters other than an underscore ' '. The different input files to COEUS and their format are described in subsequent sections.

3.5.1 Main Input

The main input file controls parameters that govern the type of analysis to be performed; convergence controls; and variables used in sensitivity, trade study, and op-timizations. There are four types of analysis in the main file that the user may select from. Selecting none will result in a simple analysis on the initial system configura-tion. These four analyses are optimization, trade study, sensitivity study, and Monte Carlo analysis. At this stage; simple analysis, trade study, and sensitivity study are fully operational while optimization is basically functional. The other parameters in this file will be described later in discussions of the types of analysis that can be performed.



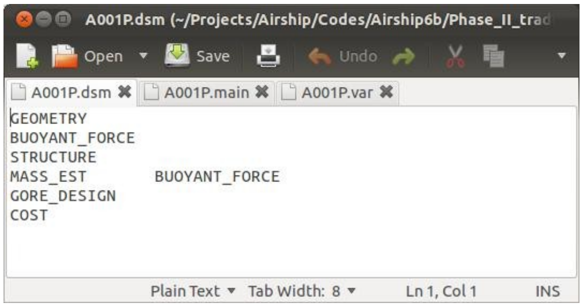
```
Convergance               0.01
MaxIterations             20
Optimization              0
TradeStudy                0
SensitivityStudy          0
MonteCarlo                0
StepAttenuation           0.4
Objective                 MAXIMIZE
Target                    40
DependentVariable         payloadMass
NumberOfIndependentVars:  1
vehicle.geometry.length      20:5.0:100.0
```

Figure 3.3: Example of the main input file to COEUS

3.5.2 Process Control Input

The process control file is a simplified representation of a Design Structure Matrix (DSM) [2]. This file has a list of all the modules in the analysis and dictates the order in which they are to be executed. The modules listed on each line of the file essentially represents the modules or analysis tasks represented by the diagonal boxes in a DSM.

The modules in this file are executed from start to finish. As with a traditional DSM many of the modules along the diagonal may be interdependent, requiring information passed forwards and backwards. Feed forward is simply implied by the order of the modules in the file. The current module would feed forward to the module on the next line. However, feed back is specified by the occurrence of a second module on the same line as the current module. If a line in the DSM file contains more than a single module name, the first module on that line would feed back information to the second. If the output variables from the first module change by an amount greater than the convergence criteria set in the main file, the module will feed back to the second module listed. Otherwise the process will continue. An example of this file is shown in figure 3.4.



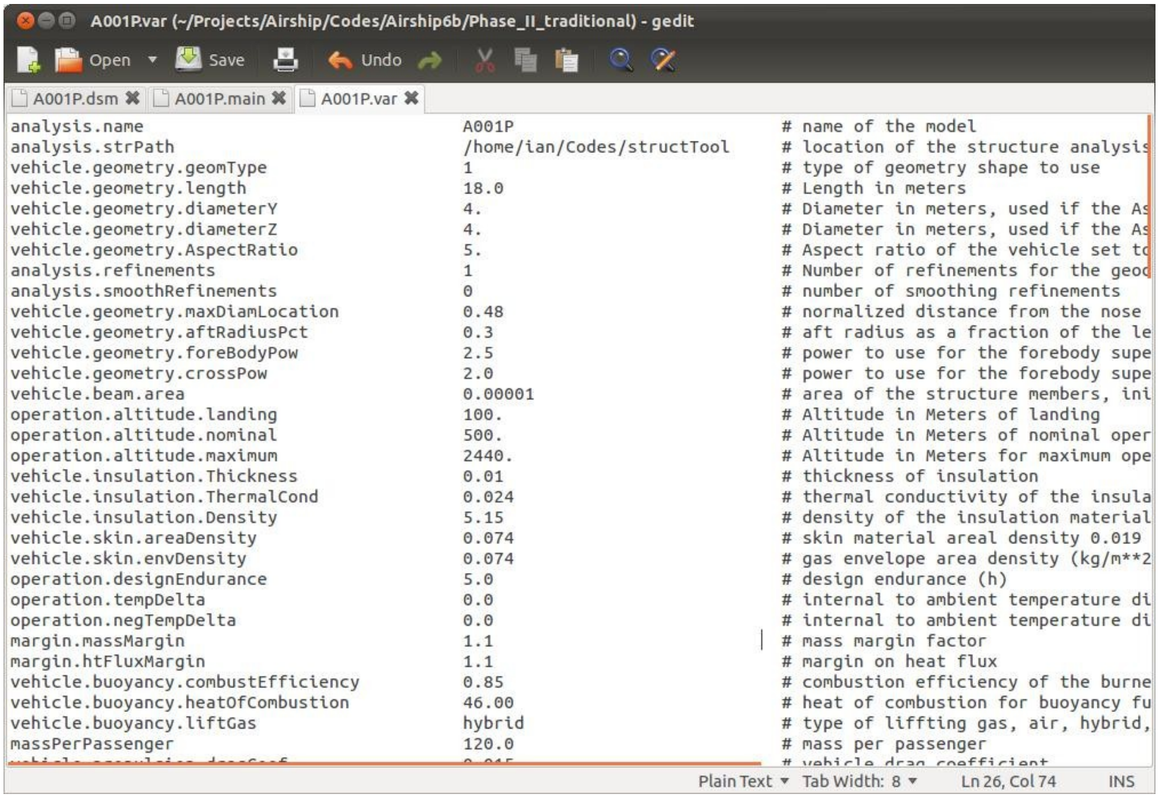
```
GEOMETRY
BUOYANT_FORCE
STRUCTURE
MASS_EST        BUOYANT_FORCE
GORE_DESIGN
COST
```

Figure 3.4: Example process control file using a simplified DSM

Here it can be seen that the BUOYANT_FORCE module is executed prior to the MASS_EST module. However, inputs to BUOYANT_FORCE are dependent on outputs from MASS_EST. Therefore, these two modules must iterate in order to converge sufficiently. The existence of the BUOYANT_FORCE module listed after the MASS_EST module directs COEUS to iterate back to BUOYANT_FORCE if the output values from MASS_EST differ from the values stored in the memory. The value in memory is the value set in the variable file or stored from a previous iteration.

3.5.3 Variable De nition Input

The variable de nition le has a .var extension and is where the user de nes and initializes all the variables in the analysis. Every variable that is used in the analysis must be in this le in order to be initialized. This le also provides an opportunity for the user to put a brief description of what each variable is, its units, or where it comes from. On each line, any information after a \#" symbol is considered to be a comment and only used for variable description.



```
analysis.name                             A001P                          # name of the model
analysis.strPath                          /home/ian/Codes/structTool     # location of the structure analysis
vehicle.geometry.geomType                 1                              # type of geometry shape to use
vehicle.geometry.length                   18.0                           # Length in meters
vehicle.geometry.diameterY                4.                             # Diameter in meters, used if the As
vehicle.geometry.diameterZ                4.                             # Diameter in meters, used if the As
vehicle.geometry.AspectRatio              5.                             # Aspect ratio of the vehicle set to
analysis.refinements                      1                              # Number of refinements for the geo
analysis.smoothRefinements                0                              # number of smoothing refinements
vehicle.geometry.maxDiamLocation          0.48                           # normalized distance from the nose
vehicle.geometry.aftRadiusPct             0.3                            # aft radius as a fraction of the le
vehicle.geometry.foreBodyPow              2.5                            # power to use for the forebody supe
vehicle.geometry.crossPow                 2.0                            # power to use for the forebody supe
vehicle.beam.area                         0.00001                        # area of the structure members, ini
operation.altitude.landing                100.                           # Altitude in Meters of landing
operation.altitude.nominal                500.                           # Altitude in Meters of nominal oper
operation.altitude.maximum                2440.                          # Altitude in Meters for maximum ope
vehicle.insulation.Thickness              0.01                           # thickness of insulation
vehicle.insulation.ThermalCond            0.024                          # thermal conductivity of the insula
vehicle.insulation.Density                5.15                           # density of the insulation material
vehicle.skin.areaDensity                  0.074                          # skin material areal density 0.019
vehicle.skin.envDensity                   0.074                          # gas envelope area density (kg/m**2
operation.designEndurance                 5.0                            # design endurance (h)
operation.tempDelta                       0.0                            # internal to ambient temperature di
operation.negTempDelta                    0.0                            # internal to ambient temperature di
margin.massMargin                         1.1                            # mass margin factor
margin.htFluxMargin                       1.1                            # margin on heat flux
vehicle.buoyancy.combustEfficiency        0.85                           # combustion efficiency of the burne
vehicle.buoyancy.heatOfCombustion         46.00                          # heat of combustion for buoyancy fu
vehicle.buoyancy.liftGas                  hybrid                         # type of liffting gas, air, hybrid,
massPerPassenger                          120.0                          # mass per passenger
vehicle.propulsion.dragCoef               0.015                          # vehicle drag coefficient
```

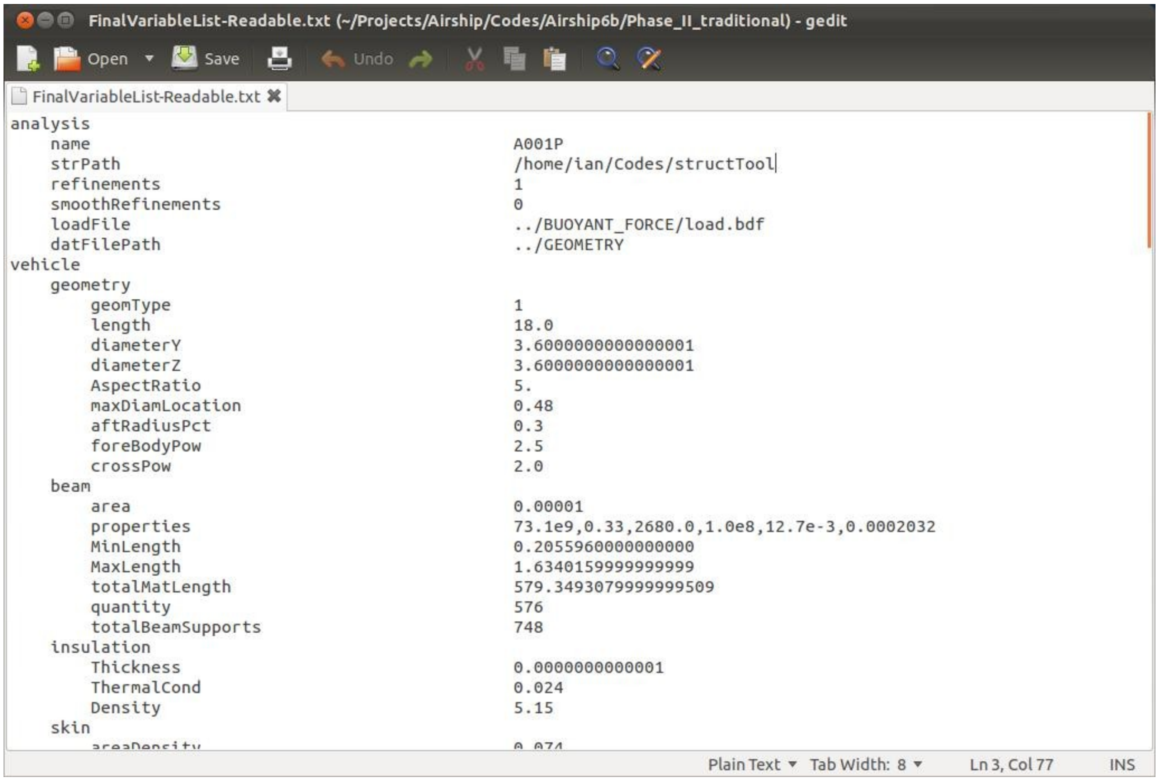
Figure 3.5: Example variable de nition le

Figure 3.5 shows an example variable input le. Every variable used in the system must be de ned here. The basic format is each line is variable name, value, and comment delimited by spaces. In order to read these variables from les during an analysis, COEUS must know what the type is for each variable whether it be integer, double precision, or string. This type is not specified in the input file; rather, it is interpreted from the initial values the user specifies in the input deck. The variable type is determined by the following; if the value does not contain a decimal \." and does not contain any non numeric or special characters then it is declared as an integer, if it is not an integer (thus containing a decimal) and only contains numeric characters then it is a double precision float, and if it cannot be qualified as either of the previous then it is declared to be a string. This was done to simplify the programming and to allow all variables to be stored in memory as string types and converted when needed to their respective types.

Variable Name

COEUS allows the user to use one of two types of naming conventions for variables. The first is a simple naming convention (e.g. length, diameter, velocity) and the second is a hierarchical or parent-child naming convention where the hierarchy of a component's relation to the parent system is depicted in its name. Using the airship example, consider a value for the volume of an airship gas envelop. the volume is an attribute of the envelope, which is a component of the buoyancy subsystem of the vehicle. Using the hierarchical convention this variable could be named vehicle.buoyancy.envelope.volume. This approach is particularly useful for models with a large number of variables. This format allows the user to group variables by variable type, subsystem, or any other decomposition approach. For instance, in the example there is an analysis group that has variables that pertain to tool paths and analysis controls. There is also a vehicle group that is decomposed further to geometry, beam, insulation, and so forth. This system allows up to 10 levels of decomposition by the user. This format is illustrated in Figure 3.6.

When the program finishes executing there are two output files written; FinalVariableList.txt and FinalVariableList-Readable.txt. The former is in the same format as the variable declaration file including the comments. The only difference in this file is that there is a single header line. In the latter file, the variables are printed in a tree structure.

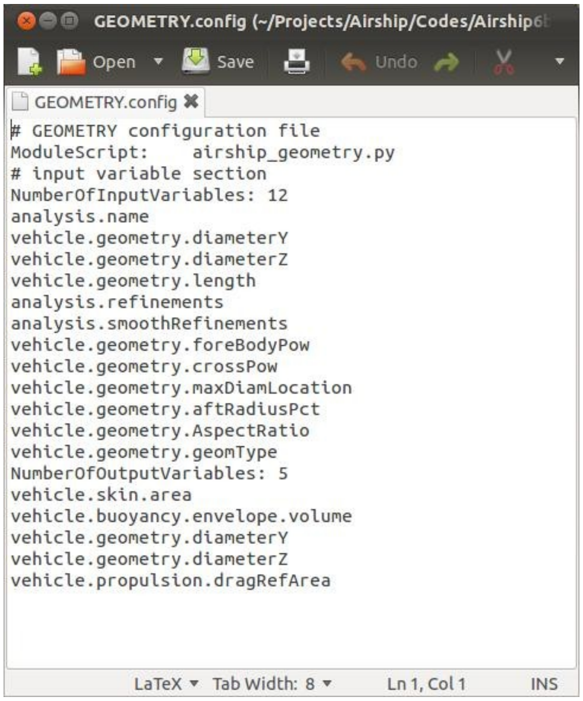

FinalVariableList-Readable.txt (~/Projects/Airship/Codes/Airship6b/Phase_II_traditional) - gedit
Open
Save
Undo
FinalVariableList-Readable.txt
Plain Text
Tab Width: 8
Ln 3, Col 77
INS

```
analysis
    name                                          A001P
    strPath                                       /home/ian/Codes/structTool
    refinements                                   1
    smoothRefinements                             0
    loadFile                                      ../BUOYANT_FORCE/load.bdf
    datFilePath                                   ../GEOMETRY
vehicle
    geometry
        geomType                                  1
        length                                    18.0
        diameterY                                 3.6000000000000001
        diameterZ                                 3.6000000000000001
        AspectRatio                               5.
        maxDiamLocation                           0.48
        aftRadiusPct                              0.3
        foreBodyPow                               2.5
        crossPow                                  2.0
    beam
        area                                      0.00001
        properties                                73.1e9,0.33,2680.0,1.0e8,12.7e-3,0.0002032
        MinLength                                 0.2055960000000000
        MaxLength                                 1.6340159999999999
        totalMatLength                            579.3493079999999509
        quantity                                  576
        totalBeamSupports                         748
    insulation
        Thickness                                 0.0000000000001
        ThermalCond                               0.024
        Density                                   5.15
    skin
        areaDensity                               0.074
```

Figure 3.6: Example of variable output in human-readable format

3.5.4 Module Configuration

The module configuration file controls what variables from the system are communicated to and from a particular module. Each module will have its own configuration file which is the only file that has some formatting to it. An example file is shown in Figure 3.7. One can see in the example that there are three sections to this file. The first is the declaration of the module program. The module is executed from COEUS by a system call. The details of module execution will be discussed in section 3.6. The second section is the variable inputs to the module followed by section three containing the output variables from the module. Both the input and output variable sections are headed by a line that declares the number of variables to read following the section header.



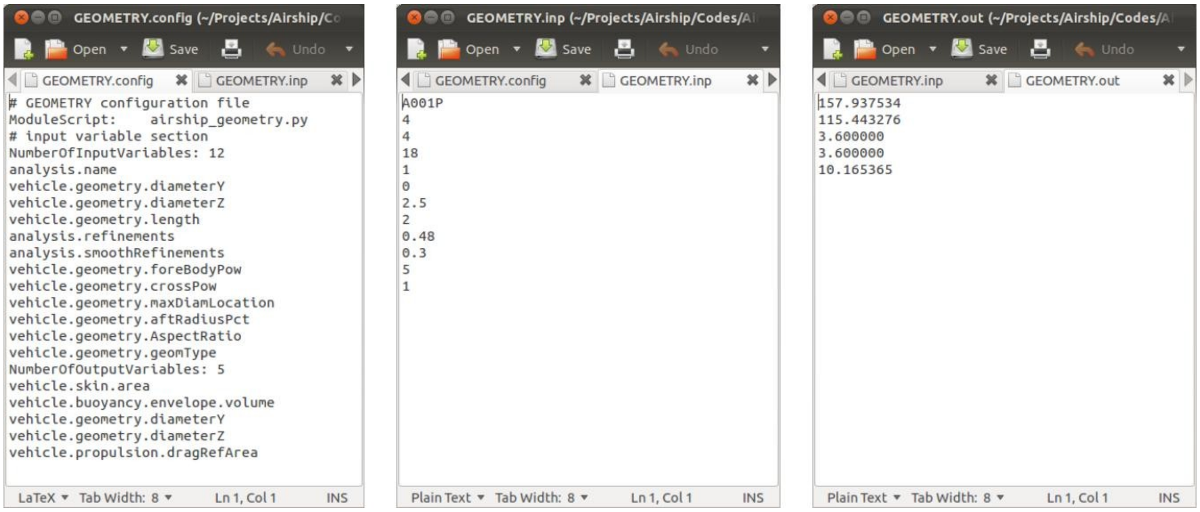
```
# GEOMETRY configuration file
ModuleScript:    airship_geometry.py
# input variable section
NumberOfInputVariables: 12
analysis.name
vehicle.geometry.diameterY
vehicle.geometry.diameterZ
vehicle.geometry.length
analysis.refinements
analysis.smoothRefinements
vehicle.geometry.foreBodyPow
vehicle.geometry.crossPow
vehicle.geometry.maxDiamLocation
vehicle.geometry.aftRadiusPct
vehicle.geometry.AspectRatio
vehicle.geometry.geomType
NumberOfOutputVariables: 5
vehicle.skin.area
vehicle.buoyancy.envelope.volume
vehicle.geometry.diameterY
vehicle.geometry.diameterZ
vehicle.propulsion.dragRefArea
```

Figure 3.7: Example a module configuration file

3.6 Module Execution

As stated in section 3.5.4, the analysis modules are essentially stand alone executables that COEUS executes via a system call. This allows the module to be written in any programming language, compiled or interpreted, and communicate with COEUS through reading and writing of files.

When a module is executed it is given two command arguments. The first is the name of the input file and the second is the output file. These input and output files contain the value of the input and output variables, respectively. The input and output files are a simple text file with one variable value listed per line. The order the values are listed is the same order the variable names are listed in the input and output sections of the module configuration file. Figure 3.8 shows a side by side comparison of the module configuration file and the module input and output files.



```
# GEOMETRY configuration file
ModuleScript:    airship_geometry.py
# input variable section
NumberOfInputVariables: 12
analysis.name
vehicle.geometry.diameterY
vehicle.geometry.diameterZ
vehicle.geometry.length
analysis.refinements
analysis.smoothRefinements
vehicle.geometry.foreBodyPow
vehicle.geometry.crossPow
vehicle.geometry.maxDiamLocation
vehicle.geometry.aftRadiusPct
vehicle.geometry.AspectRatio
vehicle.geometry.geomType
NumberOfOutputVariables: 5
vehicle.skin.area
vehicle.buoyancy.envelope.volume
vehicle.geometry.diameterY
vehicle.geometry.diameterZ
vehicle.propulsion.dragRefArea
```

```
A001P
4
4
18
1
0
2.5
2
0.48
0.3
5
1
```

```
157.937534
115.443276
3.600000
3.600000
10.165365
```

Figure 3.8: Comparison of a configuration file and its respective module input and output files. Note that the order of the variables in the configuration file corresponds to the order in the input and output files.

3.7 Program Setup Features

Setting up a set of input files is often difficult for a tool like this since it is easy to make formatting errors in the setup. This is partially addressed by the simplified and minimally formatted files used as inputs to COEUS. To address it further, COEUS has a set of commands used to generate default input files that give the general format which users can simply edit. The command line options that the user would use to generate inputs and execute a model are give in Table 3.1. The general form of the

COEUS execution command is coeus -(option) casename. The case name is chosen by the user for the model. As a general naming convention, it is best not to use any special characters other than dashes or underscores in the chosen casename.

Table 3.1: Command line options for COEUS

Option	Description
input	setup the default input files in the directory
setup	set up the directories and module config files
h	print a help message
help	print a help message
run	run the selected case
pack	pack a model into a single file for portability
unpack	unpack a model to the current directory

3.7.1 Generating main, var, and DSM Files

If a user wished to begin setting up a new model named \example1." The first step would be begin in a fresh directory. Once in this fresh directory, generic input files can be generated by running the command coeus -input example1. This will generate the files; example1.main, example1.dsm, and example1.var.

3.7.2 Generating Module Directories

The module directories are named the same as the module names found in the DSM file. The easiest way to generate these directories and related configuration files is with the -setup option. Running the command coeus -setup example1 will read in the DSM file (assuming it has already been updated with the modules the user wishes to include in the model) and create directories with the same name and a generic configuration file for each module. The user needs only to update the configuration file for each module to include the module script or program name and the input and output variable names.

3.7.3 Variable Name Declaration

In its current form, COEUS will check to ensure that every variable in the configuration files exists in the variable file at run time. If this is not the case an error is thrown and the program will exit. A future improvement that is desirable, but was not able to be implemented in this study is a command to read all the module configuration files and include any absent variables in the variable file. This would allow the user to simply enter the variable value and any comments.

3.8 Program Execution

At program execution there are several checks performed to ensure that all the variables have been declared, only one type of analysis is selected in the main file, and a check on the variable types. When COEUS runs, there is a substantial amount of information printed to the screen for the user. This information includes an echo of all the input files (main, DSM, variable, and module configuration files) and any warning or error messages. COEUS will run with warnings but will not run with errors. One common warning that could be encountered is if the user has specified a non float value be returned from a module. Due to the way convergence is computed in the presence of a feedback loop, having a return value that is an integer could lead to oscillatory behavior and never converge. Furthermore, it is obvious why convergence could not be computed if the returned variable were a string. In such an instance, a warning is issued to inform the user that these return variables are not to be used for convergence and will not influence a decision within COEUS to feed forward or feed back.

3.9 Types of Analysis

The user has 5 options for the type of analysis that may be run. Currently only four of these analyses are operational. These are a simple analysis, trade study, optimization, and sensitivity study. The analysis that is started but not fully implemented yet is Monte Carlo for uncertainty analysis.

3.9.1 Simple Analysis

In a simple analysis COEUS will execute the modules in the DSM file. In the case of a feed back loop, COEUS will iterate until convergence is achieved. Once the last module in the DSM file is executed COEUS will write the final configuration and exit.

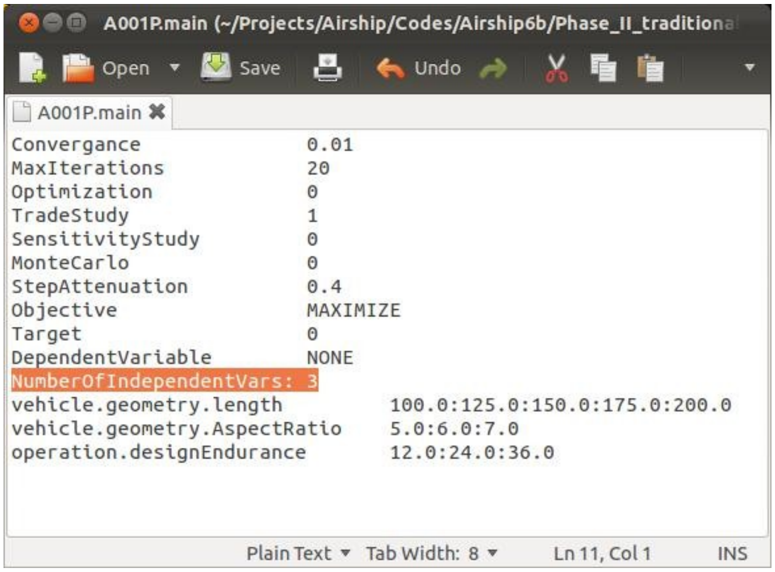
3.9.2 Trade Study Analysis

A trade study analysis is used to run a matrix of variables in different configurations. The user has the option to specify what variables are to be traded and at what values. The user may select any number of variables they wish to trade from the system. There is no limit to the number of variables or the number of states for each variable.

For each variable and value to be traded COEUS will run a simple analysis. The difference between the trade study and the sensitivity study is that in the trade study a N-dimensional matrix of cases is run. If the user is trading five variables, then a five dimensional matrix is the result. A trade study will run every combination of variable values. As can be seen, performing a trade study with many variables, beyond three, becomes more difficult to visualize and process.

Variables used in the trade study are listed in the last section of the main input file. Figure 3.9 shows an example of the trade variable declaration section in the main input file. The variable declaration section begins at the highlighted line. Here the number of variables to parse is given. Each line below this header has the variable name and a colon delimited list of values for that particular variable.

The output for the trade study is a ASCII plot file that contains all of the analyses run for all the traded variables. Each line of this file represents an analysis and each column represents a variable. An additional file that is output is the key file. This file has the same case name as the var, DSM, and main files but with a .key extension. This file contains a list of variable names and what column they are found in the result plot file.

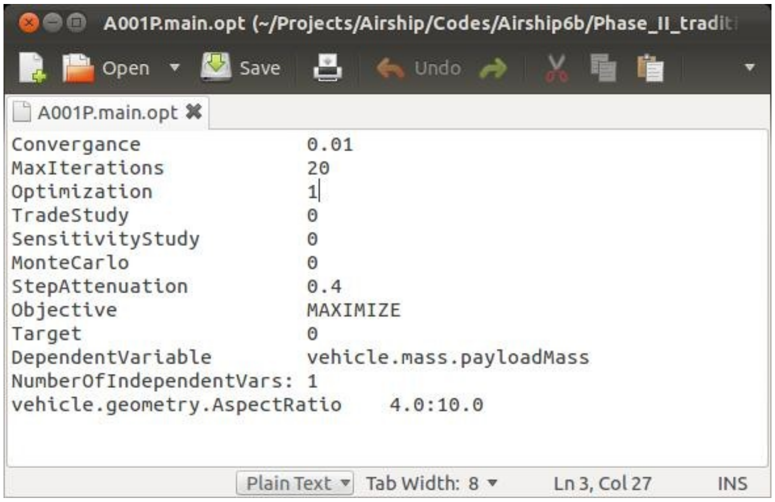

A001P.main (~/Projects/Airship/Codes/Airship6b/Phase_II_traditiona
Open
Save
Undo
A001P.main
Plain Text
Tab Width: 8
Ln 11, Col 1
INS

```
Convergance                 0.01
MaxIterations               20
Optimization                0
TradeStudy                  1
SensitivityStudy            0
MonteCarlo                  0
StepAttenuation             0.4
Objective                   MAXIMIZE
Target                      0
DependentVariable           NONE
NumberOfIndependentVars: 3
vehicle.geometry.length            100.0:125.0:150.0:175.0:200.0
vehicle.geometry.AspectRatio       5.0:6.0:7.0
operation.designEndurance          12.0:24.0:36.0
```

Figure 3.9: Example main input file illustrating the declaration of trade study variables and values.

3.9.3 Optimization Analysis

An optimization analysis allows the user to maximize or minimize a single variable within the system based upon a set of other variables specified in the main input file. Figure 3.10 shows how the main input file would be configured for an optimization analysis. Notice that the \Objective" is specified to maximize the dependent variable. In this case the dependent variable is set to the payload mass of the system. Looking at the variable declaration section of the file this maximization of the payload mass is to be based upon the aspect ratio of the vehicle (i.e. the ratio of length to diameter).


A001P.main.opt (~/Projects/Airship/Codes/Airship6b/Phase_II_tradit
Open
Save
Undo
A001P.main.opt
Plain Text
Tab Width: 8
Ln 3, Col 27
INS

```
Convergance                 0.01
MaxIterations               20
Optimization                1
TradeStudy                  0
SensitivityStudy            0
MonteCarlo                  0
StepAttenuation             0.4
Objective                   MAXIMIZE
Target                      0
DependentVariable           vehicle.mass.payloadMass
NumberOfIndependentVars:    1
vehicle.geometry.AspectRatio    4.0:10.0
```

Figure 3.10: Example main input file illustrating the declaration of optimization variables and their upper and lower bounds.

In the previous example for a trade study, the variables accompanying the variable name in the declaration section was a colon delimited list of values at which to analyze. In this case there is only two values, the lower and upper bounds.

The output files for the optimization analysis is the same for the general analysis. These are the FinalVariableList.txt and FinalVariableList-Readable.txt files. In addition to these files there is a convergence file that is written out that prints the dependent variable and each independent variable for each iteration. This can be used to track the convergence of the dependent variable.

Optimization Approach

The optimization approach used in this tool was intended to be an initial solution for simple problems with additional optimizer options to be added later. What is cur-rently available is a gradient based optimizer that works best for a single independent variable. However, the optimizer will also handle multiple variables so long as there are little to no interaction effects of the independent variables. The reason for this will be made clear shortly.

A standard approach for gradient based optimization with a single independent variable is to compute the first and second derivatives and use these to compute the step required for the first derivative to be zero, indicating the presence of an extrema. This is currently the approach taken in COEUS and works reasonably well when used for a single independent variable. However, with a system as complex as the ones intended to be tackled with COEUS, it is unlikely that only a single independent variable would be used. In this case since the independent variable is a vector the first derivative would be a vector as shown in Equation 3.1. Taking the derivative of this would not result in a vector but rather a Hessian matrix as shown in Equation 3.2. The current optimization implementation results in a Hessian matrix where only the diagonal values are non-zero. This essentially is performing the optimization on each variable independently of the others. If there were little interaction between variables this would still be a viable solution. In reality this results in an inefficient and often incorrect solution when dealing with multiple variables.

The best approach in this situation would be to use an optimization subroutine that have been optimized for performance and validated. Furthermore, for complex systems arriving at a maximum from a gradient optimizer does not guarantee a global optima. To address this including stochastic optimization capabilities like genetic algorithms would be valuable.

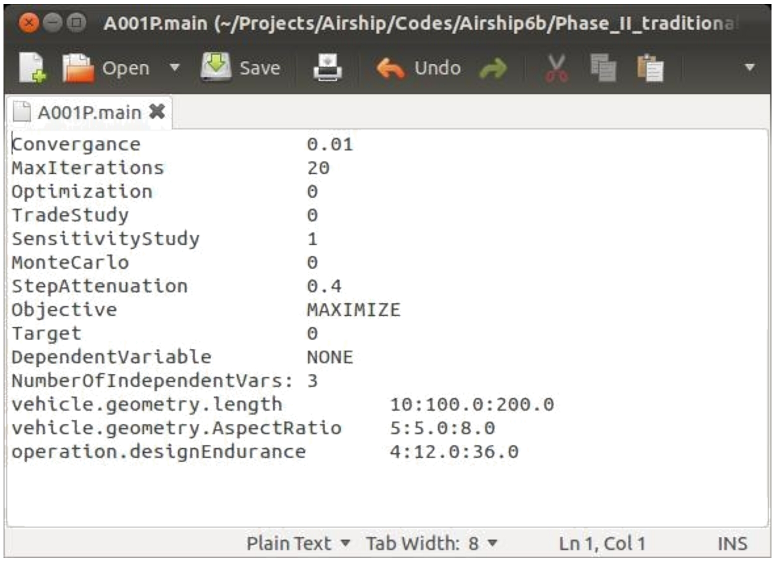
$$\nabla f(x) = g(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{3.1}$$

$$\nabla^2 f(x) = H(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \vdots & & \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{3.2}$$

3.9.4 Sensitivity Study Analysis

The sensitivity study is similar to the trade study analysis. However, as stated earlier, the trade study will run every combination of every value stated. The sensitivity study will only vary one variable at a time not combinations of variables. In essence, the sensitivity study trades one variable while leaving all others at their nominal values. If trading more than one variable, upon completion of one variable trade, its original (nominal) value is replaced before moving on to the next variable to trade.

The inputs for the sensitivity study are slightly different than the previous two analysis types. This difference is in the values given with the variable name. In this case, the value given is a colon delimited list of 3 values. The first value is an integer indicating the number of samples to run in the sensitivity for that variable. The second and third values are the lower and upper bounds of the range to be sampled for the specific variable, respectively. Currently, only uniform sampling is available. If COEUS determines that the first value in the delimited list is not an integer, but a float, the program will issue a warning to the user but will proceed after converting the given float to an integer.


A001P.main (~/Projects/Airship/Codes/Airship6b/Phase_II_traditiona
Open
Save
Undo
A001P.main
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

```
Convergance                 0.01
MaxIterations               20
Optimization                0
TradeStudy                  0
SensitivityStudy            1
MonteCarlo                  0
StepAttenuation             0.4
Objective                   MAXIMIZE
Target                      0
DependentVariable           NONE
NumberOfIndependentVars:    3
vehicle.geometry.length             10:100.0:200.0
vehicle.geometry.AspectRatio        5:5.0:8.0
operation.designEndurance           4:12.0:36.0
```

Figure 3.11: Example main input file illustrating the declaration of sensitivity variables and the sampling values.

The outputs are similar to the trade study analysis as well. However, instead of listing all the results in a single plot file, each variable for which sensitivities are computed is printed in a separate plot file. Since the order the variables are printed to the plot files is consistent only a single key file is printed in the top directory. Each of the plot files are in a separate directory using a naming convention according to the variable being traded.

3.9.5 Monte Carlo Analysis

The Monte Carlo analysis is not operational at this time in the code. This would be a valuable asset to have and, as such, will be left to future work. The input will likely be very similar to the sensitivity study in that the lower and upper bounds will be specified as well as the number of samples to take over that region. In addition, there will be inputs that govern the type of distribution by which to sample and any values required to drive the shape of the sampling distribution. Due to the complexity of many of the models that COEUS can handle and without modifications that would allow execution on distributed systems or super computers, COEUS would most likely be set up to use a Latin Hyper Cube approach rather than a brute force Monte Carlo.

3.10 COEUS Output Files

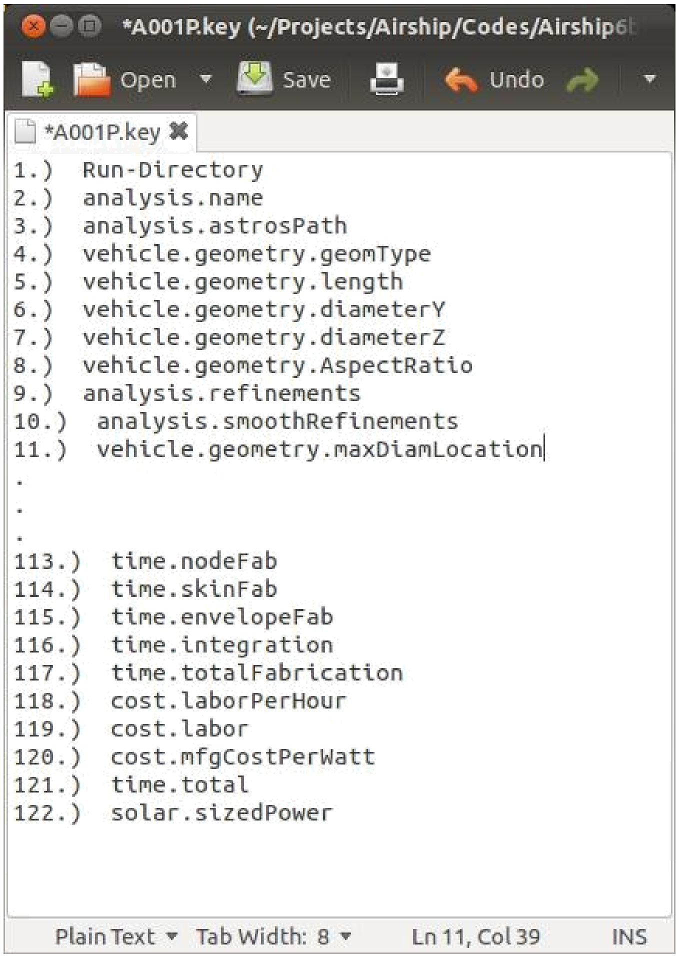
For a general analysis, COEUS outputs the system variables at their final converged state. As mentioned previously in section 3.5.3, this is done in two forms. The first is identical to the variable declaration file with the addition of a text header line. The second is the more human readable format that was Illustrated in Figure 3.6. These output files are present for any type of analysis that is selected. However, it only has significance when running a simple analysis or an optimization. If running a trade study or sensitivity study where many different cases are run, only the last run configuration is written to these files. The outputs for these other analyses will be discussed in subsequent sections.

3.10.1 Key File Format

The key file is written out for both the trade study and the sensitivity analyses. This file tells the user what variable is located in what column of the result plot file. An example key file is shown in Figure 3.12.

3.10.2 Plot File Output Format

As stated previously, the plot output format is used for the trade study and sensitivity analyses. Each line in this file represents a different analysis case. And each column represents a different variable of the system. This format allows the user to access the



```
1.)  Run-Directory
2.)  analysis.name
3.)  analysis.astrosPath
4.)  vehicle.geometry.geomType
5.)  vehicle.geometry.length
6.)  vehicle.geometry.diameterY
7.)  vehicle.geometry.diameterZ
8.)  vehicle.geometry.AspectRatio
9.)  analysis.refinements
10.)  analysis.smoothRefinements
11.)  vehicle.geometry.maxDiamLocation
.
.
.
113.)  time.nodeFab
114.)  time.skinFab
115.)  time.envelopeFab
116.)  time.integration
117.)  time.totalFabrication
118.)  cost.laborPerHour
119.)  cost.labor
120.)  cost.mfgCostPerWatt
121.)  time.total
122.)  solar.sizedPower
```

Figure 3.12: Example key file

result from all of the cases run in a format that makes plotting easier by consolidating all the results in a single file. The general format of the file is illustrated in Table 3.2

Table 3.2: Plot File General Format

case1_var1	case1_var2	case1_var3	case1_var4	...	case1_varN
case2_var1	case2_var2	case2_var3	case2_var4	...	case2_varN
case3_var1	case3_var2	case3_var3	case3_var4	...	case3_varN
case4_var1	case4_var2	case4_var3	case4_var4	...	case4_varN
...	...	...	...	...	...
caseN_var1	caseN_var2	caseN_var3	caseN_var4	...	caseN_varN

Chapter 4

Example Application

The example that will be used to demonstrate the capabilities of COEUS will be an airship model. This model is composed of a the following modules; parametric geometry generation, buoyancy and forces, structure, subsystem mass estimation, skin gore design, and manufacturing cost estimation. These modules and their relationship are illustrated in the design structure matrix shown in Figure 4.1. These modules will be discussed in more detail in subsequent sections.

4.1 Parametric Geometry

The parametric geometry module, named GEOMETRY in the analysis, is a script wrapper to a shape generation tool written in C++ that uses inputs from the user to generate a finite element representation of the vehicle skin and structure. This FE model is used in all subsequent modules. The GEOMETRY module also computes the volume, surface area, and drag reference area of the vehicle for subsequent use. Area calculations are performed by calculating the surface area of each triangle using the vector cross product of two sides as shown in Equation 4.1. In this equation vectors

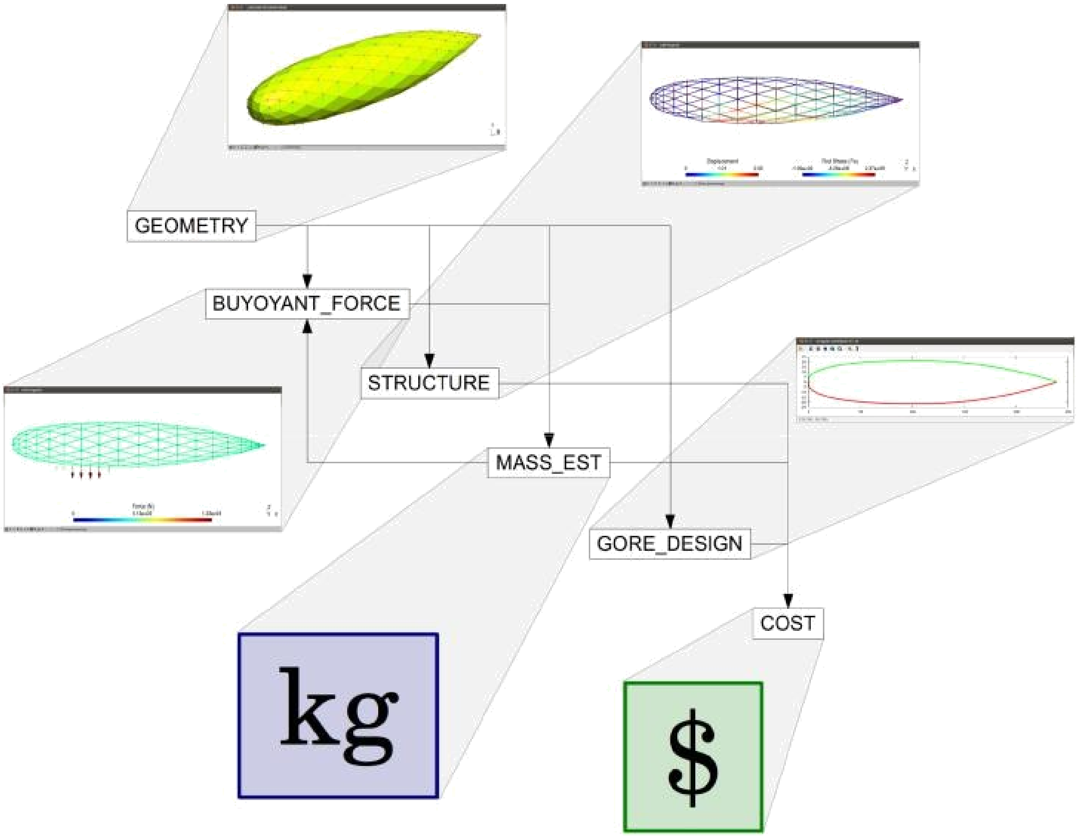


Figure 4.1: Design structure matrix and representative outputs for the example model

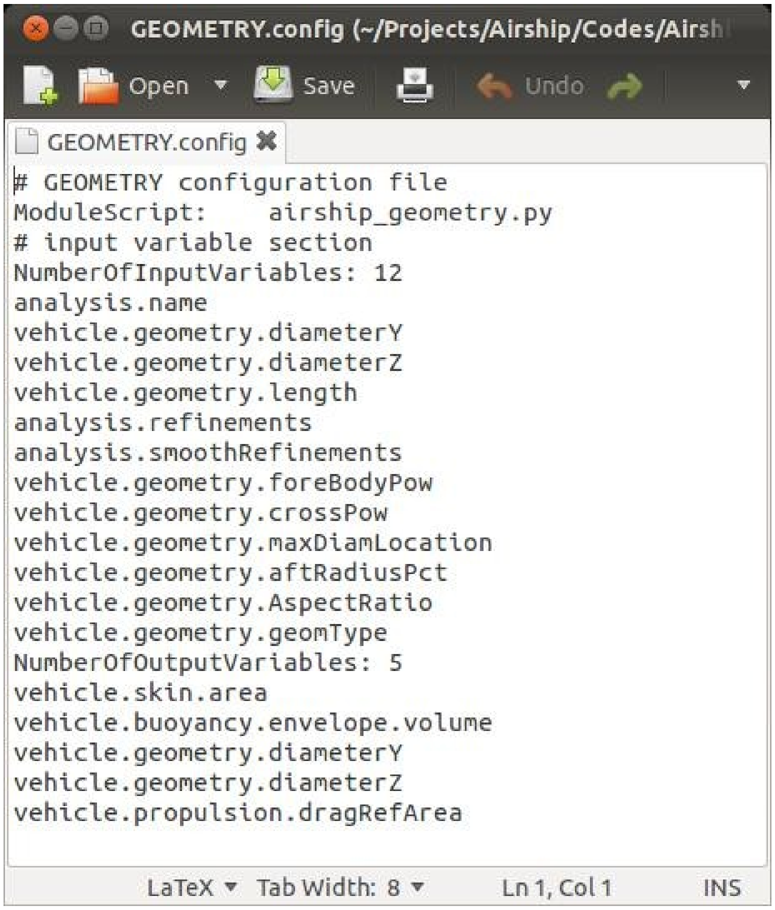
$\tilde{A}$ and $\tilde{B}$ represent two sides of the triangular element and A_{tri} is the resulting area of the triangular element.

$$A_{tri} = \frac{|\tilde{A}\ \tilde{B}|}{2} \tag{4.1}$$

The volume of the vehicle is calculated using the divergence theorem. This theo-rem is shown in its general form in Equation 4.2. In the case of calculating the volume of a discrete shell, the divergence theorem takes the form of Equation 4.3. Here V is the volume of the enclosed region, A_i is the area of each discrete element, x_{cent_i} is the x coordinate of the centroid of the triangular element, $n\hat{}_i$ is the unit normal vector of the triangular element, and $x\hat{}_{axis}$ is a unit vector in the x direction.

$$ZZ_S \; d\tilde{S} \;\; \tilde{F} \tag{4.2}$$

$$V = \overset{n}{\underset{=0}{X_i}} (A_i \;\; x_{cent_i} \;\; n\hat{}_i \;\; x\hat{}_{axis}) \tag{4.3}$$



```
# GEOMETRY configuration file
ModuleScript:    airship_geometry.py
# input variable section
NumberOfInputVariables: 12
analysis.name
vehicle.geometry.diameterY
vehicle.geometry.diameterZ
vehicle.geometry.length
analysis.refinements
analysis.smoothRefinements
vehicle.geometry.foreBodyPow
vehicle.geometry.crossPow
vehicle.geometry.maxDiamLocation
vehicle.geometry.aftRadiusPct
vehicle.geometry.AspectRatio
vehicle.geometry.geomType
NumberOfOutputVariables: 5
vehicle.skin.area
vehicle.buoyancy.envelope.volume
vehicle.geometry.diameterY
vehicle.geometry.diameterZ
vehicle.propulsion.dragRefArea
```

Figure 4.2: Module Configuration for the GEOMETRY module.

The geometry generation tool mentioned above takes inputs in a very simple format like most of the modules. An example of the input is shown below. The inputs are passed to this tool as stdin on the command line; therefore, everything after the \#" character is ignored and is provided simply for the understanding of the user. This is the file written out by the GEOMETRY module. The generated geometry is comprised of line elements and triangular surface elements. The line elements represent the beams of the structure and the surface elements represent the skin or OML of the vehicle. An example of the resulting geometry is shown in Figure 4.3

```
1                        # Enter The   number of Bar Refinement Iterations
0                        # Enter the   number of Smoothing Refinement Iterations
2.500000                 # Enter the   power to use for the forbody ellipse
0.450000                 # Enter the   location of the max diameter
0.100000                 # Enter the   aft radius
1                        # Enter the   number of values on the following line
2.000000                 # Enter the   power distribution for the super ellipse
239.000000               # Enter the   nominal major diameter (x)
42.500000                # Enter the   nominal horizontal minor diameter (y)
42.500000                # Enter the   nominal vertical minor diameter (z)
A001P                    # Provide  a  name for the   output    model to  be  written to
0                        # 0 for no stl, 1 for stl
0                        # Enter 1   to  transpose x  and z    access or  0  not too
1                        # 1 to generate surfaces and 0 to not
1.0                      # vehicle Scale parameter
1                        # vehicle   geometry   type
1                        # vehicle   geometry   smoothing
```

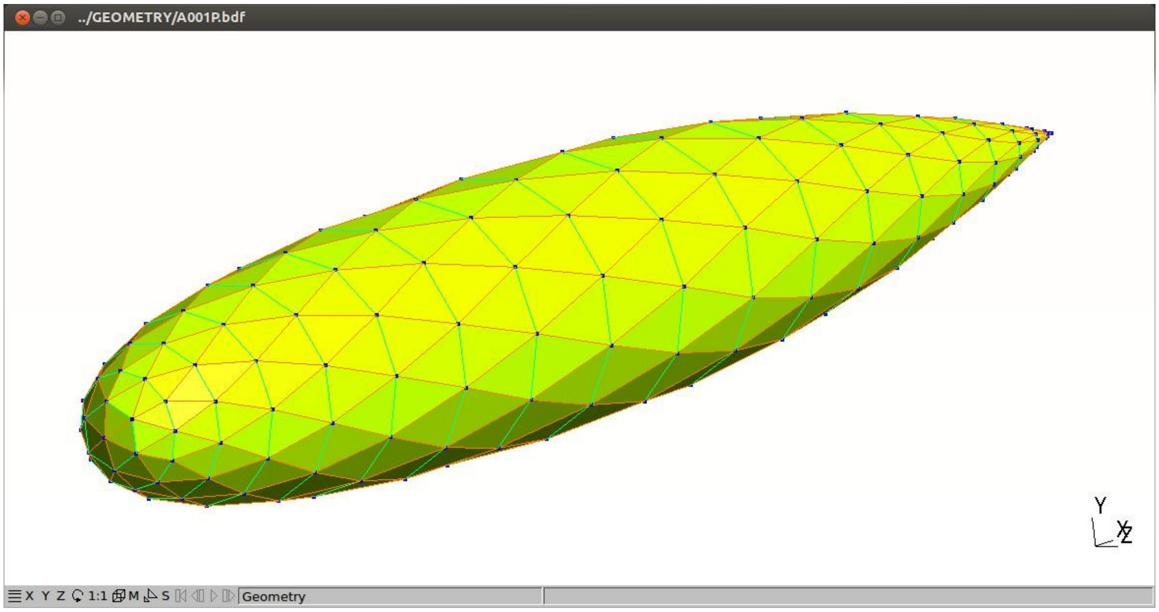


Figure 4.3: Geometry output from parametric geometry generation tool

4.2 Buoyancy

This module is named BUOYANT FORCE and computes data related to the buoy-ancy of the vehicle. This module is also written in Python. The input and output variables to this module are listed and described in Tables 6.3 and 6.4, respectively. The computations performed here are to determine the lift of the system in the stated environment and prepare data for the structure module. The primary functions of this module are to compute:

the ambient environment pressure, temperature, and density from the opera-tional altitude.

the pressure on each panel relative to the ships keel based on the hydrostatic equation given in Equation 4.4 where P is the pressure at the panel center, P_0 is the reference pressure at the keel of the vehicle, is the local density, g is the gravitational constant of 9.81 $m=s^2$, and h is the height of the panel above the keel.

the total lift of the system based on the hydrostatic pressure distribution.

the node forces resulting from the pressure on each panel. These forces are written to a file that is used by the structure module, described later.

the center of lift based on the pressure distribution.

4.2.1 Ambient Environment

The ambient environment of the airship (pressure, temperature, and density) at the operational altitude is determined from a table of gas states versus the altitude from a standard atmosphere table. The nominal altitude of operation, which is defined to be at the keel of the ship, is specified by the user and is an input to this module. The table lookup routine works by scanning the altitudes in the table. Once the routine finds a line in the table that is below the specified value and the following line is above the specified value. The requested properties (pressure, temperature, and density) are found using a linear interpolation.

4.2.2 Surface Pressure

The surface pressure on every element in the structure is determined using the hy-drostatic equation shown in Equation 4.4. This equation assumes that the change in density from the top of the vehicle to the bottom of the vehicle does not change appreciably. In equation 4.4; P is the local pressure, P_0 is the pressure at the keel of the ship, is the density, g is the gravitational constant of 9.81 m=s, and h is the height of the panel in question above the keel.

$$P = P_0 + \ gh \tag{4.4}$$

4.2.3 Total System Lift

The total lift of the system is the summation of all the buoyancy pressure forces on the vehicle. This force can be depicted by the summation in Equation 4.5. Here P_i is the pressure, assumed uniform over the element, for the i^{th} element, A_i is the area of the i^{th} element, $\hat{n}_i$ is the normal vector of the i^{th} element, and $\hat{k}$ is the vertical or YAW axis of the vehicle.

$$L_{tot} = \sum_{i}^{n} P_i A_i (\hat{n}_i \ \hat{k}) \tag{4.5}$$

Since each element is triangular, the area can be found by taking two sides of the triangle to be vectors, call them V_{ai} and V_{bi} . The area of each element can then be found using Equation 4.6. Similarly, once the dot product is found, the unit normal vector of the element is computed by Equation 4.7.

$$A_i = \frac{\tilde{V}_{ai} \tilde{V}_{bi}}{2} \qquad (4.6)$$

$$\hat{n}_i = \frac{\tilde{V}_{a_i} \tilde{V}_{b_i}}{\tilde{V}_{ai} \tilde{V}_{bi}} \qquad (4.7)$$

4.2.4 Nodal Forces

The forces on the nodes are derived from the pressure loads on the elements. The resulting force on the elements can be found as a product of the pressure and area, acting in the normal direction, as discussed above. For each element, the resultant force is divided by three and applied to each node. From the node perspective, this would result in up to 6 different load vectors applied since each force is acting in the normal direction. To determine a single load and direction, the vector sum of all loads on a node are reported. These values are written to a file in a NASTRAN Bulk Data Format for use by the structure module to be discussed later.

4.2.5 Center of Lift

The center of lift is determined by summing the moment about the nose of the vehicle. This uses the discrete lift contributed by each element and is set equal to the total lift of the vehicle multiplied by some lever arm. This lever length is the center of lift. This approach is depicted in Equation 4.8 where X_{cl} is the location of the center of lift from the nose, L_i is the lift of the i_{th} element, x_i is the distance along the X axis from the nose for the i_{th} element, and L_{tot} is the total buoyant lift of the system. This value is returned as a dimension in meters.

$$X_{cl} = \frac{P_{i}^{n} (L_i \quad x_i)}{L_{tot}} \tag{4.8}$$

4.3 Structure

The STRUCTURE module is special in that it performs analysis at a higher fidelity than the others. Its purpose is to determine the mass of the structure as well as the configuration of all the beams in the structure. The beam configuration is in reference to the thickness of the beam material and the number of supports that are required. The STRUCTURE module completes this task through direct simulation with a NASTRAN like finite element tool. The module takes the geometry input and loads generated in previous modules and writes out the finite element model. With additional work, this has the potential to be run at a very high fidelity, including modal and buckling analysis. However, at this point the solution utilizes idealized elements. While the actual beams are composed of a tri-member, truss-like structure the beam elements in the analysis are taken to be ideal line elements. The finite element tool is called twice in this analysis. The first time determines the force imparted on the beam. As all the beams in the analysis and in reality are straight, there are only axial and gravity loads acting on the beams. At this point, load contributions from gravity are assumed to be negligible and ignored. After the first run the axial loads on every beam are known. Each beam is then sized analytically such that it can withstand the axial loads and resist buckling. Essentially, the total area of the three components of the truss should be such that the beam should not fail due to yielding and the truss should be wide enough where the area moment of inertia is sufficient to withstand static buckling.

While, in practice, this approach would have the deficiency if sizing dynamic components by static methods, this approach is solely intended as a proof of concept. This is a demonstration that high fidelity analysis can be automated and included in this type of system level study.

The results of this module are handled in two ways. Some information like structure mass, total beam length, maximum and minimum beam length, etc are written out and fed back into COEUS for use by modules downstream. Other information like the beam configuration details discussed earlier are written to a file. This information is not needed by COEUS; furthermore, it would be too difficult to feed that quantity of information back in a meaningful way. Currently, this information is both written to a data file as well as a plot file. Below is an excerpt from data file and the plot file presenting the beam stress and displacement is shown in Figure 4.4.

```
i , * * * W , N , mass ,    length , bLen ,         si , force
   1 1.490895 E -05  * *     * 32  0.598045       44 .9 0 28 13   5. 678891    9   .6287 E -01 in        -1419.900000
   2 1.797642 E -05  * *     * 27  0.674101       41 .9 7 66 73   5. 878877    1  .1610 E +00 in        -1712.040000
   3 1.485729 E -04  * *     * 2  2.492354      1 8.7 78 3 10    5.678892   9.5954 E +00 in         -14149.800000
   4 5.889198 E -05  * *     * 7  1.259856      2 3.9 47 0 12    5.878877   3.8034 E +00 in         -5608.760000
   5 1.485729 E -04  * *     * 2  2.492353      1 8.7 78 3 09    5.678892   9.5954 E +00 in         -14149.800000
   6 5.889198 E -05  * *     * 7  1.259856      2 3.9 47 0 11    5.878876   3.8034 E +00 in         -5608.760000
   7 1.491063 E -05  * *     * 32  0.598113       44 .9 0 28 13   5. 678891    9   .6298 E -01 in        -1420.060000
   8 1.797716 E -05  * *     * 27  0.674128       41 .9 7 66 73   5. 878877    1  .1610 E +00 in        -1712.110000
   9 1.634819 E -04  * *     * 2  2.742455      1 8.7 78 3 08    5.678892   1.0558 E +01 in    1 5 5 6 9. 7 0 0 0 0 0
  10 7.686704 E -05  * *     * 5  1.520585      2 2. 14 40 4 6    5.878877   4.9643 E +00 in    7 3 2 0 .6 7 0 0 0 0
  11 1.634840 E -04  * *     * 2  2.742491      1 8. 77 83 0 9    5.678892   1.0558 E +01 in    1 5 5 6 9. 9 0 0 0 0 0
  12 7.686725 E -05  * *     * 5  1.520589      2 2. 14 40 4 7    5.878877   4.9644 E +00 in    7 3 2 0 .6 9 0 0 0 0
  13 7.108448 E -05  * *     * 9  2.334226      3 6. 75 82 3 5    8.392129   4.5909 E +00 in    6 7 6 9 .9 5 0 0 0 0
  14 1.008632 E -04  * *     * 8  4.116257      4 5. 68 31 5 9    1 0  .8 08 12 6  6.5141 E +00 in 9 6 0 6. 0 2 0 0 0 0
  15 8.839152 E -05  * *     * 7  2.699302      3 4. 18 43 4 4    8.392093   5.7086 E +00 in         -8418.240000
  16 8.851479 E -05  * *     * 9  3.743373      4 7. 34 05 9 8    1 0  .8 08 14 6  5.7166 E +00 in        -8429.980000
  17 4.710101 E -05  * *     * 8  0.866213      2 0. 58 64 1 7    4.870516   3.0419 E +00 in    4 4 8 5 .8 1 0 0 0 0
  18 4.076573 E -05  * *     * 9  0.776902      2 1. 33 32 7 7    4.870517   2.6328 E +00 in    3 8 8 2 .4 5 0 0 0 0
  19 8.257022 E -06  * *     * 88  1.021028       1 3 8 . 4 2 0 5 3 7 8.392093         5.3327 E -01 in      -786.383000
  20 1.778900 E -05  * *     * 52  1.884838       1 1 8 . 6 0 6 6 8 6 10 . 80 81 4 6  1.1489 E +00 in       -1694.190000
  21 4.220990 E -05  * *     * 16  1.725727       4 5.  76 6 14  4   8. 392093      2.7261 E +00 in        -4019.990000
  22 3.397590 E -05  * *     * 27  2.342337       7 7.  17 2 91  2   10    .8 0 81 46  2.1943 E +00 in       -3235.800000
  23 5.449427 E -05  * *     * 6  0.929462      1 9. 09 27 0 2    4.870517   3.5194 E +00 in    5 1 8 9 .9 3 0 0 0 0
  24 5.449542 E -05  * *     * 6  0.929482      1 9. 09 26 9 7    4.870516   3.5195 E +00 in    5 1 9 0 .0 4 0 0 0 0
  25 4.221137 E -05  * *     * 16  1.725789       4 5.  76 6 19  7   8. 392102      2.7262 E +00 in        -4020.130000

   *
   *
   *
```

```
628 3.870960 E -06     *  *  * 43   0  .150907   43 .6 3 93 41  10. 27 4 88 4  2.5000 E -01 in  13   .1 89 5 00
629 3.870960 E -06     *  *  * 213  1.170117  3 3 8 . 3 7 3 9 8 9   1 0 .2 74 8 86  2.5000 E -01 in  3 0 9 . 6 1 3 0 0 0
630 3.870960 E -06     *  *  * 24   0  .036604  10 .5 8 49 92  2.146473    2.5000 E -01 in   1 0 1. 5 3 1 0 0 0
631 3.492983 E -05     *  *  * 10   0  .654784  20 .9 8 40 04  4.628728    2.2559 E +00 in  3 3 26 . 6 5 0 0 0 0
632 7.208250 E -06     *  *  * 5      0.013485 2.094077  0.555943         4.6553 E -01 in  -686.500000
633 3.756165 E -05     *  *  * 9      0.680303 2 0   .2 74 22 0 4.628728    2.4259 E +00 in   3 5 7 7. 3 0 0 0 0 0
634 6.919196 E -05     *  *  * 4      1.033816 1 6   .7 25 30 8 4.628727    4.4687 E +00 in   6 5 8 9. 7 1 0 0 0 0
635 1.227650 E -05     *  *  * 2      0.020161 1.838306  0.555936         7.9286 E -01 in  -1169.190000
636 6.919206 E -05     *  *  * 4      1.033818 1 6   .7 25 30 9 4.628727    4.4687 E +00 in   6 5 8 9. 7 2 0 0 0 0
637 3.756081 E -05     *  *  * 9      0.680288 2 0   .2 74 21 7 4.628727    2.4258 E +00 in   3 5 7 7. 2 2 0 0 0 0
638 7.207599 E -06     *  *  * 5      0.013483 2.094059  0.555938         4.6549 E -01 in  -686.438000
639 3.492899 E -05     *  *  * 10   0  .654768  20 .9 8 39 99  4.628727    2.2558 E +00 in  3 3 26 . 5 7 0 0 0 0
640 3.333236 E -05     *  *  * 11   0  .645974  21 .6 9 37 82  4.628727    2.1527 E +00 in        -3174.510000
641 6.143487 E -06     *  *  * 6      0.011960 2.179307  0.555938         3.9677 E -01 in  5 8 5 . 09 4 0 0 0
642 3.589100 E -05     *  *  * 10   0  .672802  20 .9 8 40 00  4.628727    2.3180 E +00 in        -3418.190000
643 7.245998 E -05     *  *  * 4      1.082644 16 .7 25 30 8 4.628727    4.6797 E +00 in       -6900.950000
644 1.440506 E -05     *  *  * 2      0.023656 1.838306  0.555936         9.3033 E -01 in  1 3 7 1 .9 1 0 0 0 0
645 7.245977 E -05     *  *  * 4      1.082641 16 .7 25 30 9 4.628727    4.6797 E +00 in       -6900.930000
646 3.589078 E -05     *  *  * 10   0  .672798  20 .9 8 40 00  4.628727    2.3180 E +00 in        -3418.170000
647 6.143812 E -06     *  *  * 6      0.011961 2.179307  0.555938         3.9679 E -01 in  5 8 5 . 12 5 0 0 0
648 3.333183 E -05     *  *  * 11     0.645963  21 .6 9 37 81  4.628727    2.1527 E +00 in        -3174.460000
```

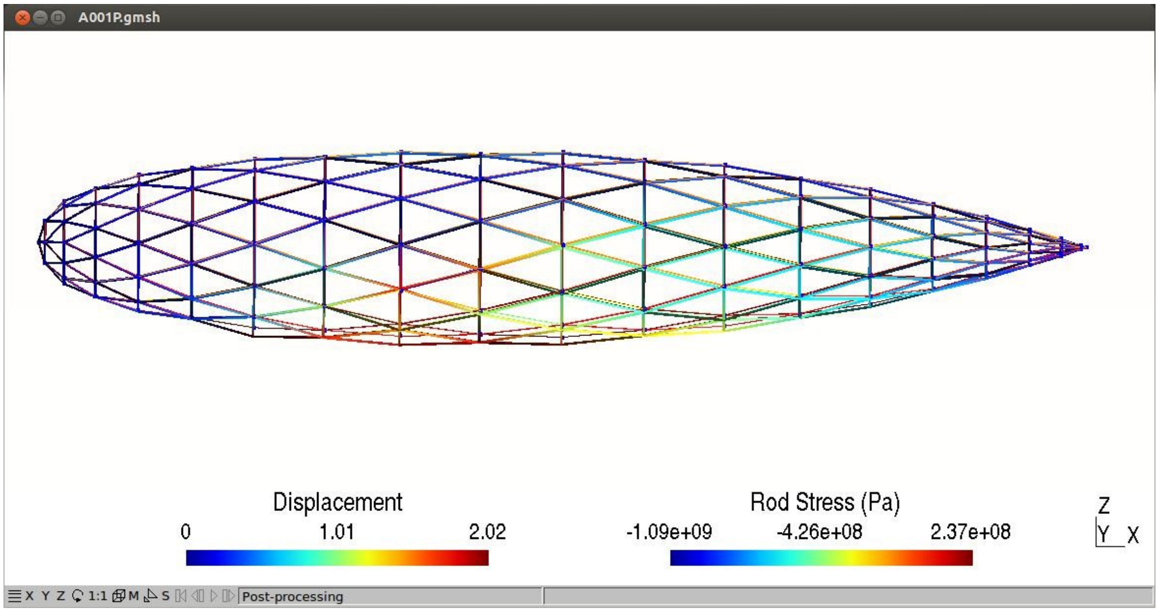


Figure 4.4: Structural module plot illustrating displacement and beam stress

4.4 Subsystem Mass

The MASS_EST module is responsible for computing the masses of components and subsystems that is not determined by direct simulation. This module uses a low -

fidelity approach where mass of each component is computed using a mass estimating relation or MER. These MERs are often determined using parametric equations de-veloped from existing vehicles or components that have been developed in the past. In this case, the MERs were developed using a simplified calculation from computed parameters. For example, to compute the skin mass, The area of the vehicle skin is known reasonable well from the finite element model. Therefore, the mass can be approximated reasonably well by multiplying the skin area by an areal mass of the skin material that is specified by the user.

Equations 4.9 through 4.17 describe a selection of mass estimating relations from the MASS EST module. These equations are provided to give a sense of the low fidelity approach in this module.

$$m_{bF\ uel} = S_{skin}\ E\ K_{ins}\ H_c\ \frac{(T_{in}\ T_{out})}{t_{ins\ comb}}\ M_{mass} \tag{4.9}$$

$$m_{bT\ ank} = m_{bF\ uel}\ tmf \tag{4.10}$$

$$m_{ins} = t_{ins}\ S_{skin}\ (_{ins}\ \ _{out})\ M_{mass} \tag{4.11}$$

$$m_{skin} = S_{skin}\ _{skin}\ M_{mass} \tag{4.12}$$

$$m_{fin} = (m_{str} + m_{skin})\ f\ mf \tag{4.13}$$

$$h_{loss} = \frac{K_{ins}\ S_{skin}\ (T_{in}\ \ T_{out})}{t_{ins}} \tag{4.14}$$

$$F_{drag} = C_d\ q_1\ S_{drag} \tag{4.15}$$

$$P_{prop} = F_{drag}\ v_1 \tag{4.16}$$

$$m_{pF\ uel} = \frac{p_{prop}\ \ BSF\ C\ E}{2{:}2} \tag{4.17}$$

4.5 Skin Gore Design

The GORE_DESIGN module designs the gores that make up the vehicle skin. The outputs of the module give the number of gores required, gore length and width, as well as an additional plot file that contains the actual gore coordinates that could be fed into a CAD program or plotter.

The module takes information from the user, such as gore material width, gore overlap, and the maximum vehicle geometry. With this information the module computes the number of gores that will use the width of the supplied material best. Based on the a contour file that is written out by the geometry tool, gore width as a function of its length is computed and written to the plot file. Below is an excerpt from that file.

```
 X              R            S              WIDTH       HALF  WIDTH
 0  .0000      -0.0000      0  .0000        0.0200        0.0100
 0  .0590      -1.5208      1  .5219       -0.0693       -0.0347
 0  .2358      -2.6462      2  .6612       -0.1354       -0.0677
 0  .5303      -3.6565      3  .7136       -0.1947       -0.0974
 0  .9423      -4.5964      4  .7398       -0.2499       -0.1250
 1  .4712      -5.4850      5  .7738       -0.3021       -0.1510
 2  .1167      -6.3325      6  .8392       -0.3519       -0.1759
 2  .8779      -7.1453      7  .9527       -0.3996       -0.1998
 3  .7543      -7.9272      9  .1273       -0.4455       -0.2227
 4  .7449      -8.6812     10  .3721       -0.4898       -0.2449
 5  .8487      -9.4091     11  .6944       -0.5325       -0.2663
 7  .0647     -10.1124     13  .0991       -0.5738       -0.2869
 8  .3917     -10.7919     14  .5899       -0.6137       -0.3069
 9  .8283     -11.4484     16  .1694       -0.6523       -0.3261
11  .3731     -12.0825     17  .8393       -0.6895       -0.3447
13  .0247     -12.6944     19  .6006       -0.7254       -0.3627
14  .7813     -13.2844     21  .4537       -0.7601       -0.3800
16  .6413     -13.8528     23  .3985       -0.7935       -0.3967
18  .6028     -14.3995     25  .4348       -0.8256       -0.4128
20  .6638     -14.9246     27  .5617       -0.8564       -0.4282
22  .8224     -15.4282     29  .7782       -0.8860       -0.4430
25  .0764     -15.9102     32  .0832       -0.9143       -0.4571
27  .4236     -16.3706     34  .4751       -0.9413       -0.4707
29  .8617     -16.8095     36  .9524       -0.9671       -0.4835
32  .3882     -17.2267     39  .5131       -0.9916       -0.4958
35  .0007     -17.6223     42  .1554       -1.0148       -0.5074
37  .6966     -17.9963     44  .8771       -1.0368       -0.5184
40  .4732     -18.3487     47  .6760       -1.0575       -0.5287
43  .3278     -18.6795     50  .5497       -1.0769       -0.5384
46  .2575     -18.9887     53  .4957       -1.0950       -0.5475
49  .2596     -19.2765     56  .5115       -1.1119       -0.5560
52  .3310     -19.5430     59  .5944       -1.1276       -0.5638
55  .4686     -19.7884     62  .7417       -1.1420       -0.5710
58  .6695     -20.0128     65  .9504       -1.1552       -0.5776
```

```
 61  .9303       -20.2166    69  .2176        -1.1671       -0.5836
 65  .2480       -20.4000    72  .5404        -1.1779       -0.5890
 68  .6193       -20.5634    75  .9156        -1.1875       -0.5938
 72  .0407       -20.7072    79  .3400        -1.1960       -0.5980

  *
  *
  *

213  .9234        -7.5069   222  .4087        -0.4208       -0.2104
216  .1774        -6.9079   224  .7409        -0.3856       -0.1928
218  .3360        -6.3216   226  .9778        -0.3512       -0.1756
220  .3970        -5.7502   229  .1166        -0.3177       -0.1588
222  .3585        -5.1958   231  .1549        -0.2851       -0.1426
224  .2185        -4.6606   233  .0903        -0.2537       -0.1268
225  .9752        -4.1466   234  .9206        -0.2235       -0.1117
227  .6267        -3.6557   236  .6436        -0.1947       -0.0973
229  .1716        -3.1898   238  .2572        -0.1673       -0.0837
230  .6082        -2.7507   239  .7594        -0.1415       -0.0708
231  .9352        -2.3401   241  .1485        -0.1174       -0.0587
233  .1512        -1.9595   242  .4226        -0.0951       -0.0475
234  .2550        -1.6106   243  .5803        -0.0746       -0.0373
235  .2456        -1.2946   244  .6201        -0.0560       -0.0280
236  .1220        -1.0128   245  .5406        -0.0395       -0.0197
236  .8833        -0.7662   246  .3409        -0.0250       -0.0125
237  .5287        -0.5559   247  .0197        -0.0126       -0.0063
238  .0577        -0.3827   247  .5763        -0.0025       -0.0012
238  .4696        -0.2473   248  .0099         0.0055        0.0027
238  .7642        -0.1502   248  .3201         0.0112        0.0056
238  .9410        -0.0912   248  .5065         0.0146        0.0073
239  .0000        -0.0000   248  .6151         0.0200        0.0100
```

4.6 Cost Estimation

The current costing model has capability of calculating rough fabrication costs including both materials and labor costs. Both use a similar approach to the MERs discussed in Section 4.4. What is different in this instance is that there was some unit of measure defined and this unit of measure (UOM) has either a time or monetary value assigned to it. From this UOM the total amounts at the system and subsystem levels can be aggregated.

In the case of labor, the total time required to perform specific tasks are computed and a general labor cost per hour is applied to translate the value into a monetary value. While this works fine for a proof of concept, in actuality one would want to assign labor hourly costs by task or task type.

To illustrate the approach here, Equation 4.18 computes the time to fabricate all the beams of the system. This equation is based on the fact that each beam has 2 ends and each support structure requires 6 attachments. The 8.0 in the denominator is simply a conversion from hours to days.

$$T_{beamFab} = \frac{2N_{beams}T_{end} + 6N_{sup}T_{joint}}{8{:}0} M_{time} \tag{4.18}$$

4.7 COEUS Analysis Examples

In the following sections a number of examples will be presented that demonstrate how the different types of analysis COEUS is capable of can be used in a design environment. A simple analysis, trade study, and sensitivity study will be performed to come up with a reasonable understanding of the affects of a few variables on the system. The simple analysis will present an initial baseline configuration or benchmark. The sensitivity study will identify how modifying certain variables effect the performance of the system. And finally, a trade study of the same variables used in the sensitivity study will be used to illustrate how the interaction affects of varying different variables simultaneously can influence the systems overall performance.

4.7.1 Example Simple Analysis

In this case, a simple analysis will be performed on a starting configuration. This will demonstrate the ability of COEUS to perform an analysis on a fixed configura-tion. In this instance, a fixed configuration means that the input variables are fixed. However, there are output variables that are to be determined. This example will demonstrate how COEUS tracks convergence of these output variables and iterates on the different modules until the vehicle is closed, where closure of the vehicle is defined as convergence of the output variables.

The section below shows the results of the analysis that are printed to screen. This informs the user of what is happening during the run. The information echoed to the user is, first, the name and version of the tool. Second, is definitions of the modules listed in the order they are to be executed. Third, is the configuration information if the main, var, and dsm files. Fourth, is the run information. This section contains all of the convergence information for the various modules as well as which module is being executed at that moment. Prior to the actual execution COEUS checks if the case setup is configured correctly. These checks include type declarations and ensuring that all variables used in modules are declared in the variable file. If not, the beginning of this section will contain error and/or warning messages.

```
COEUS (COncurrent Engineering Utility for Systems)
POC: Ian.Dupzyk@gmail.com
Version 1.4.2


ANALYSIS MODULES
    1 GEOMETRY--> NONE
    2 BUOYANT_FORCE --> NONE
    3 STRUCTURE --> NONE
    4 MASS_EST --> BUOYANT_FOR
    CE 5 GORE_DESIGN --> NONE
    6 COST --> NONE



MODULE_CONFIGURATION: GEOMETRY
    Module Script:    airship_geometry.py
    Input Variables:  12
        analysis.name
        vehicle.geometry.diameterY
        vehicle.geometry.diameterZ
        vehicle.geometry.length
        analysis.refinements
        analysis.smoothRefinements
        vehicle.geometry.foreBodyPow
        vehicle.geometry.crossPow
        vehicle.geometry.maxDiamLocation
        vehicle.geometry.aftRadiusPct
        vehicle.geometry.AspectRatio
        vehicle.geometry.geomType
    Output Variables: 5
        vehicle.skin.area
        vehicle.buoyancy.envelope.volume
        vehicle.geometry.diameterY
        vehicle.geometry.diameterZ
        vehicle.propulsion.dragRefArea


MODULE_CONFIGURATION: BUOYANT_FORCE
    Module Script:      airship_buoyancy.py
    Input Variables:    13
        analysis.name
        operation.altitude.nominal
        operation.altitude.maximum
        operation.altitude.landing
        operation.tempDelta
        operation.negTempDelta
        vehicle.buoyancy.envelope.volume
        vehicle.buoyancy.liftGas
        vehicle.mass.dryMass
        vehicle.buoyancy.HeOffsetMassFraction
        solar.solarPanelNu
        solar.surfSolarFlux
        solar.sizedPower
    Output Variables: 14
        vehicle.buoyancy.netLift
        vehicle.buoyancy.liftPosition
        environment.pAmbient
        environment.tempOut
        vehicle.internalEnvironments.tempIn
        vehicle.internalEnvironments.rhoIn
        environment.rhoOut
        vehicle.buoyancy.volHeAlt
        vehicle.buoyancy.volHeSTP
        solar.electricPower
        vehicle.buoyancy.envelope.lift
        vehicle.buoyancy.HeOffsetMassFraction
        vehicle.buoyancy.volHeLand
        vehicle.buoyancy.netLiftLand


MODULE_CONFIGURATION: STRUCTURE
    Module Script:      airship_structure.py
    Input Variables:    17
        analysis.name
        analysis.strPath
        vehicle.beam.properties
        vehicle.skin.properties
        analysis.loadFile
        vehicle.buoyancy.envelope.lift
        vehicle.gondola.bounds
        vehicle.buoyancy.liftPosition
        vehicle.geometry.length
        vehicle.geometry.diameterZ
        vehicle.mass.structMass
        vehicle.beam.area
        margin.massMargin
        vehicle.geometry.geomType
        vehicle.mass.insulationMass
        vehicle.mass.skinMass
        vehicle.mass.HeEnvMass
      Output Variables: 2
          vehicle.mass.structMass
          vehicle.beam.totalBeamSupports


MODULE_CONFIGURATION: MASS_EST
    Module     Script:  /home/ian/Projects/Airship/Codes/Airship6b/bin/airship_massEst.py
    Input      Variables:  33
    analysis . name
    vehicle . skin . area
    vehicle . buoyancy . envelope . volume
    vehicle . beam . area
    vehicle . beam . properties
    vehicle . insulation . ThermalCond
    vehicle . insulation . Density
    vehicle . skin . areaDensity
    vehicle . skin . envDensity
    vehicle . buoyancy . netLift
    vehicle . buoyancy . netLiftLand
    margin . massMargin
    operation . designEndurance
    operation . range
    vehicle . internalEnvironments . tempIn
    environment . tempOut
    vehicle . buoyancy . heatOfCombustion
    vehicle . buoyancy . combustEfficiency
    massPerPassenger
    vehicle . insulation . Thickness
    vehicle . finMassFraction
    margin . htFluxMargin
    vehicle . buoyancy . liftGas
    environment . velocity
    vehicle . propulsion . dragRefArea
    vehicle . propulsion . dragCoef
    environment . rhoOut
    vehicle . propulsion . bsfc
    vehicle . propulsion . numberOfEngines
    vehicle . mass . structMass
    vehicle . mass . node . unitMass
    solar . electricPower
    solar . panelSpecificMass
Output Variables: 32
    vehicle . mass . skinMass
    vehicle . mass . insulationMass
    vehicle . mass . payloadMass
    vehicle . mass . buoyancyFuelMass
    vehicle . passengers
    vehicle . internalEnvironments . tempInF
    vehicle . htFlux
    vehicle . beam . MinLength
    vehicle . beam . MaxLength
    vehicle . mass . finMass
    vehicle . beam . quantity
    vehicle . beam . totalMatLength
    vehicle . mass . burnerMass
    vehicle . mass . buoyancyFuelTankMass
    vehicle . mass . dryMass
    vehicle . mass . HeEnvMass
    vehicle . propulsion . propPower
    vehicle . mass . propulsion . propFuelMass
    vehicle . mass . propulsion . engineMass
    vehicle . mass . propulsion . totalEngineMass
    vehicle . node . quantity
    vehicle . mass . node . totalMass
    vehicle . mass . solarPanelMass
    vehicle . performance . energyPerTonKm
    vehicle . performance . energyPerKm
    vehicle . mass . requiredBalast
    vehicle . mass . batteryPack
    cost . batteryPack
    vehicle . battery . cellQuantity
    operation . designEndurance
```

```
                operation . range
                environment . velocity


        MODULE_CONFIGURATION: GORE_DESIGN
            Module   Script :        airship_gores.py
            Input    Variables :     5
                analysis . name
                analysis . datFilePath
                vehicle . skin . goreMaxWidth
                vehicle . skin . goreOverlap
                vehicle . geometry . diameterY
            Output   Variables : 3
                vehicle . skin . numGores
                vehicle . skin . goreWidth
                vehicle . skin . goreLength


        MODULE_CONFIGURATION: COST
Module                 Script :        airship_fabCost.py
            Input    Variables :     20
                vehicle . node . quantity
                vehicle . beam . totalMatLength
                vehicle . beam . properties
                vehicle . skin . area
                vehicle . buoyancy . volHeSTP
                cost . costPerNode
                cost . costPerBeamMaterial
                cost . costPerSkinSqM
                cost . solarCostPerWatt
                cost . costPerEnvSqM
                cost . liftGasPCM
                cost . costMargin
                solar . electricPower
                cost . beamRivetCost
                vehicle . beam . quantity
                vehicle . node . quantity
                vehicle . beam . totalBeamSupports
                cost . laborPerHour
                vehicle . skin . numGores
                vehicle . skin . goreLength
            Output   Variables : 16
                cost . totalNodeCost
                cost . totalBeamCost
                cost . skinCost
                cost . envelopeCost
                cost . liftGas
                cost . solarCost
                cost . totalRivetCost
                cost . labor
                cost . totalCost
                time . beamFab
                time . nodeFab
                time . skinFab
                time . envelopeFab
                time . integration
                time . total
                cost . mfgCostPerWatt

        MAIN FILE  CONFIGURATION

            Convergence     0.01
            Max Iterations  20
            Optimization    0
```

```
    TradeStudy        0
    SensitivityStudy  0
    MonteCarlo        0


INITIAL  VEHICLE  CONFIGURATION:

VARIABLE  LIST

    VarName : analysis . name                                Value : A001P
    VarName : analysis . strPath                             Value : / home / ian / Codes / s ...
    VarName : vehicle . geometry . geomType                  Value : 1
    VarName : vehicle . geometry . length                    Value : 239.0
    VarName : vehicle . geometry . diameterY                 Value : 42.5
    VarName : vehicle . geometry . diameterZ                 Value : 42.5
    VarName : vehicle . geometry . AspectRatio               Value :   -5.
    VarName : analysis . refinements                         Value : 1
    VarName : analysis . smoothRefinements                   Value : 0
    VarName : vehicle . geometry . maxDiamLocation           Value : 0.45
    VarName : vehicle . geometry . aftRadiusPct              Value : 0.1
    VarName : vehicle . geometry . foreBodyPow               Value : 2.5
    VarName : vehicle . geometry . crossPow                  Value : 2.0
    VarName : vehicle . beam . area                          Value : 0.00001
    VarName : operation . altitude . landing                 Value : 100.
    VarName : operation . altitude . nominal                 Value : 1524.
    VarName : operation . altitude . maximum                 Value : 1524.
    VarName : operation . range                              Value : 2088.0
    VarName : vehicle . insulation . Thickness               Value : 0.0000001
    VarName : vehicle . insulation . ThermalCond             Value : 0.0169
    VarName : vehicle . insulation . Density                 Value : 0.0000001
    VarName : vehicle . skin . areaDensity                   Value : 0.074
    VarName : vehicle . skin . envDensity                    Value : 0.074
    VarName : operation . designEndurance                    Value :   -24.00
    VarName : operation . tempDelta                          Value : 0.0
    VarName : operation . negTempDelta                       Value : 0.0
    VarName : margin . massMargin                            Value : 1.2
    VarName : margin . htFluxMargin                          Value : 1.1
    VarName : vehicle . buoyancy . combustEfficiency         Value : 0.85
    VarName : vehicle . buoyancy . heatOfCombustion          Value : 46.00
    VarName : vehicle . buoyancy . liftGas                   Value : hybrid
    VarName : massPerPassenger                               Value : 120.0
    VarName : vehicle . propulsion . dragCoef                Value : 0.015
    VarName : vehicle . propulsion . bsfc                    Value : 0.45
    VarName : vehicle . finMassFraction                      Value : 0.0
    VarName : solar . surfSolarFlux                          Value : 1000.0
    VarName : solar . panelSpecificMass                      Value : 1.8
    VarName : solar . solarPanelNu                           Value : 0.10
    VarName : vehicle . propulsion . numberOfEngines         Value : 0
    VarName : environment . velocity                         Value : 87.0
    VarName : vehicle . mass . node . unitMass               Value : 0.025
    VarName : cost . costPerNode                             Value : 3.0
    VarName : cost . costPerBeamMaterial                     Value : 9.55
    VarName : cost . solarCostPerWatt                        Value : 2.00
    VarName : cost . costPerSkinSqM                          Value : 6.10
    VarName : cost . costPerEnvSqM                           Value : 5.05
    VarName : cost . totalNodeCost                           Value : 0.00
    VarName : cost . totalBeamCost                           Value : 0.00
    VarName : cost . skinCost                                Value : 0.00
    VarName : cost . envelopeCost                            Value : 0.00
    VarName : cost . totalCost                               Value : 0.00
    VarName : cost . costMargin                              Value : 1.1
    VarName : cost . liftGas                                 Value : 0.0
    VarName : cost . liftGasPCM                              Value : 19.35
VarName :  cost . solarCost                                      Value :    0.0
VarName :  vehicle . beam . properties                           Value :  73.1e9 ,0.33 ,2680....
VarName :  vehicle . skin . properties                           Value :  112.0e9 ,0.36 ,1440...
VarName :  analysis . loadFile                                   Value :  ../BUOYANT_FORCE/...
VarName :  vehicle . gondola . bounds                            Value :    0.8
VarName :  vehicle . buoyancy . HeOffsetMassFraction             Value :  20.0
VarName :  vehicle . mass . propulsion . engineMass              Value :  0.0
VarName :  vehicle . mass . propulsion . totalEngineMass         Value :    0.0
VarName :  solar . electricPower                                 Value :    0.0
VarName :  vehicle . mass . solarPanelMass                       Value :  0.0
VarName :  vehicle . propulsion . propPower                      Value :    0.0
VarName :  vehicle . mass . propulsion . propFuelMass            Value :    0.0
VarName :  vehicle . propulsion . dragRefArea                    Value :    0.0
VarName :  vehicle . buoyancy . volHeAlt                         Value :    0.0
VarName :  vehicle . buoyancy . volHeSTP                         Value :    0.0
VarName :  vehicle . buoyancy . volHeLand                        Value :    0.0
VarName :  vehicle . skin . area                                 Value :  0.0
VarName :  analysis . datFilePath                                Value :  ../ GEOMETRY
VarName :  vehicle . skin . goreMaxWidth                         Value :  1.27
VarName :  vehicle . skin . goreOverlap                          Value :  0.01
VarName :  vehicle . skin . goreLength                           Value :    0.0
VarName :  vehicle . skin . numGores                             Value :    0
VarName :  vehicle . skin . goreWidth                            Value :    0.0
VarName :  vehicle . buoyancy . envelope . volume                Value :    0.0
VarName :  vehicle . internalEnvironments . rhoIn                Value :  0.0
VarName :  environment . rhoOut                                  Value :    0.0
VarName :  vehicle . buoyancy . netLift                          Value :  0.0
VarName :  vehicle . buoyancy . netLiftLand                      Value :    0.0
VarName :  vehicle . buoyancy . envelope . lift                  Value :    0.0
VarName :  vehicle . buoyancy . liftPosition                     Value :    0.0
VarName :  vehicle . mass . requiredBalast                       Value :  0.0
VarName :  vehicle . mass . dryMass                              Value :  0.0
VarName :  vehicle . mass . skinMass                             Value :    0.0
VarName :  vehicle . mass . HeEnvMass                            Value :    0.0
VarName :  vehicle . mass . insulationMass                       Value :  0.0
VarName :  vehicle . mass . structMass                           Value :    0.0
VarName :  vehicle . mass . node . totalMass                     Value :  0.0
VarName :  vehicle . mass . burnerMass                           Value :    0.0
VarName :  vehicle . mass . finMass                              Value :  0.0
VarName :  vehicle . mass . payloadMass                          Value :  0.0
VarName :  environment . tempOut                                 Value :    0.0
VarName :  vehicle . internalEnvironments . tempIn               Value :    0.0
VarName :  vehicle . internalEnvironments . tempInF              Value :    0.0
VarName :  environment . pAmbient                                Value :  0.0
VarName :  vehicle . mass . buoyancyFuelMass                     Value :    0.0
VarName :  vehicle . mass . buoyancyFuelTankMass                 Value :    0.0
VarName :  vehicle . passengers                                  Value :    0
VarName :  vehicle . beam . MinLength                            Value :    0.0
VarName :  vehicle . beam . MaxLength                            Value :    0.0
VarName :  vehicle . beam . totalMatLength                       Value :  0.0
VarName :  vehicle . beam . quantity                             Value :    0
VarName :  vehicle . node . quantity                             Value :    0
VarName :  vehicle . htFlux                                      Value :    0.0
VarName :  vehicle . performance . energyPerTonKm                Value :  0.0
VarName :  vehicle . performance . energyPerKm                   Value :    0.0
VarName :  cost . beamRivetCost                                  Value :  0.00
VarName :  cost . totalRivetCost                                 Value :    0.0
VarName :  vehicle . beam . totalBeamSupports                    Value :    0
VarName :  time . beamFab                                        Value :    0.0
VarName :  time . nodeFab                                        Value :    0.0
VarName :  time . skinFab                                        Value :    0.0
VarName :  time . envelopeFab                                    Value :    0.0
VarName :  time . integration                                    Value :    0.0
VarName :  time . totalFabrication                               Value :    0.0
```

```
    VarName : cost . laborPerHour                          Value : 0.0
    VarName : cost . labor                                 Value : 0.0
    VarName : cost . mfgCostPerWatt                        Value :   0.0
    VarName : time . total                                 Value : 0.0
    VarName : solar . sizedPower                           Value :   0.0
    VarName : vehicle . mass . batteryPack                 Value : 0.0
    VarName : cost . batteryPack                           Value :   0.0
    VarName : vehicle . battery . cellQuantity             Value :   0

<   WARNING >>> MODULE OUTPUT VARIABLE " vehicle . beam . totalBeamSupports " IS NOT OF TYPE DOUBLE
    check the declaration in the case . var file , double type should contain a decimal
    This variable will not be used in determining convergance of a feedback module
<   WARNING >>> MODULE OUTPUT VARIABLE " vehicle . passengers " IS NOT OF TYPE DOUBLE check
    the declaration in the case . var file , double type should contain a decimal This variable will not be used in d
    etermining convergance of a feedback module
<   WARNING >>> MODULE OUTPUT VARIABLE " vehicle . beam . quantity " IS NOT OF TYPE DOUBLE check the
    declaration in the case . var file , double type should contain a decimal This variable will not be used in det
    ermining convergance of a feedback module
<   WARNING >>> MODULE OUTPUT VARIABLE " vehicle . node . quantity " IS NOT OF TYPE DOUBLE check the
    declaration in the case . var file , double type should contain a decimal This variable will not be used in det
    ermining convergance of a feedback module
<   WARNING >>> MODULE OUTPUT VARIABLE " vehicle . battery . cellQuantity " IS NOT OF TYPE DOUBLE check
    the declaration in the case . var file , double type should contain a decimal
    This variable will not be used in determining convergance of a feedback module
<   WARNING >>> MODULE OUTPUT VARIABLE " vehicle . skin . numGores " IS NOT OF TYPE DOUBLE check
    the declaration in the case . var file , double type should contain a decimal This variable will not be used in d
    etermining convergance of a feedback module

MODULE : GEOMETRY ITERATION : 1

MODULE : BUOYANT_FORCE ITERATION : 1

MODULE : STRUCTURE  ITERATION : 1

MODULE : MASS_EST ITERATION : 1
    VARIABLE : vehicle . mass . skinMass          <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . mass . insulatio ...     <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . mass . payloadMass       <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . internalEnviro ...       <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . beam . MinLength         <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . beam . MaxLength         <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . beam . totalMatL ...     <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . mass . dryMass           <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . mass . HeEnvMass         <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . propulsion . pro ...     <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . mass . propulsio ...     <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . mass . node . tota ...   <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . performance . en ...     <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . performance . en ...     <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . mass . batteryPack       <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : cost . batteryPack                 <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : operation . designEndurance        <<< CONVERGENCE NOT ACHIEVED >>>

MODULE : BUOYANT_FORCE ITERATION : 2

<   MODULE CONVERGED IN 2 ITERATIONS >>>

MODULE : STRUCTURE ITERATION : 1

MODULE : MASS_EST ITERATION : 1
    VARIABLE : vehicle . mass . payloadMass       <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . mass . dryMass           <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . performance . en ...     <<< CONVERGENCE NOT ACHIEVED >>>
    VARIABLE : vehicle . mass . requiredB ...     <<< CONVERGENCE NOT ACHIEVED >>>
```

```
MODULE : BUOYANT_FORCE ITERATION : 2

<     MODULE CONVERGED IN 2 ITERATIONS >>>

MODULE : STRUCTURE ITERATION : 1

MODULE : MASS_EST ITERATION : 1
     VARIABLE : vehicle . mass . payloadMass      <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . dryMass          <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . performance . en ...    <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . requiredB ...    <<< CONVERGENCE NOT ACHIEVED >>>

MODULE : BUOYANT_FORCE ITERATION : 2

<     MODULE CONVERGED IN 2 ITERATIONS >>>

MODULE : STRUCTURE ITERATION : 1

MODULE : MASS_EST ITERATION : 1
     VARIABLE : vehicle . mass . payloadMass      <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . dryMass          <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . performance . en ...    <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . requiredB ...    <<< CONVERGENCE NOT ACHIEVED >>>

MODULE : BUOYANT_FORCE ITERATION : 2

<     MODULE CONVERGED IN 2 ITERATIONS >>>

MODULE : STRUCTURE ITERATION : 1

MODULE : MASS_EST ITERATION :  1
     VARIABLE : vehicle . mass . payloadMass      <<<CONVERGENCE NOT ACHIEVED  >>>
     VARIABLE : vehicle . performance . en ...    <<<CONVERGENCE NOT ACHIEVED  >>>
     VARIABLE : vehicle . mass . requiredB ...    <<<CONVERGENCE NOT ACHIEVED  >>>

MODULE : BUOYANT_FORCE ITERATION : 2

<<< MODULE CONVERGED IN 2 ITERATIONS >>>

MODULE : STRUCTURE ITERATION :   1

MODULE : MASS_EST ITERATION :  1

MODULE : GORE_DESIGN ITERATION : 1

MODULE : COST ITERATION : 1


FINAL VEHICLE CONFIGURATION :

VARIABLE LIST

     VarName : analysis . name                                Value : A001P
     VarName : analysis . strPath                             Value : / home / ian / Codes / s ...
     VarName : vehicle . geometry . geomType                  Value :   1
     VarName : vehicle . geometry . length                    Value : 239.0
     VarName : vehicle . geometry . diameterY                 Value : 42.5000000000000000
     VarName : vehicle . geometry . diameterZ                 Value : 42.5000000000000000
     VarName : vehicle . geometry . AspectRatio               Value : -5.
     VarName : analysis . refinements                         Value :   1
     VarName : analysis . smoothRefinements                   Value :   0
     VarName : vehicle . geometry . maxDiamLocation           Value : 0.45
     VarName : vehicle . geometry . aftRadiusPct              Value :   0.1
     VarName :  vehicle . geometry . foreBodyPow                   Value :   2.5
     VarName :  vehicle . geometry . crossPow                      Value :   2.0
     VarName :  vehicle . beam . area                              Value :  0.00001
     VarName :  operation . altitude . landing                     Value :   100.
     VarName :  operation . altitude . nominal                     Value :   1524.
     VarName :  operation . altitude . maximum                     Value :   1524.
     VarName :  operation . range                                  Value : 2088.000000000000...
     VarName :  vehicle . insulation . Thickness                   Value :  0.0000001
     VarName :  vehicle . insulation . ThermalCond                 Value :  0.0169
     VarName :  vehicle . insulation . Density                     Value :  0.0000001
     VarName :  vehicle . skin . areaDensity                       Value :  0.074
     VarName :  vehicle . skin . envDensity                        Value :  0.074
     VarName :  operation . designEndurance                        Value : 24.0000000000000000
     VarName :  operation . tempDelta                              Value :   0.0
     VarName :  operation . negTempDelta                           Value :   0.0
     VarName :  margin . massMargin                                Value :  1.2
     VarName :  margin . htFluxMargin                              Value :   1.1
     VarName :  vehicle . buoyancy . combustEfficiency             Value :  0.85
     VarName :  vehicle . buoyancy . heatOfCombustion              Value :  46.00
     VarName :  vehicle . buoyancy . liftGas                       Value :  hybrid
     VarName :  massPerPassenger                                   Value :  120.0
     VarName :  vehicle . propulsion . dragCoef                    Value :  0.015
     VarName :  vehicle . propulsion . bsfc                        Value :  0.45
     VarName :  vehicle . finMassFraction                          Value :   0.0
     VarName :  solar . surfSolarFlux                              Value :  1000.0
     VarName :  solar . panelSpecificMass                          Value :   1.8
     VarName :  solar . solarPanelNu                               Value :  0.10
     VarName :  vehicle . propulsion . numberOfEngines             Value :   0
     VarName :  environment . velocity                             Value : 87.0000000000000000
     VarName :  vehicle . mass . node . unitMass                   Value :  0.025
     VarName :  cost . costPerNode                                 Value :   3.0
     VarName :  cost . costPerBeamMaterial                         Value :  9.55
     VarName :  cost . solarCostPerWatt                            Value :  2.00
     VarName :  cost . costPerSkinSqM                              Value :  6.10
     VarName :  cost . costPerEnvSqM                               Value :  5.05
     VarName :  cost . totalNodeCost                               Value : 719.3999999999999773
     VarName :  cost . totalBeamCost                               Value : 505.5464440000000081
     VarName :  cost . skinCost                                    Value : 160473.0050000000...
     VarName :  cost . envelopeCost                                Value : 132850.6025000000...
     VarName :  cost . totalCost                                   Value : 3370294.908944000...
     VarName :  cost . costMargin                                  Value :   1.1
     VarName :  cost . liftGas                                     Value : 3075746.354999999...
     VarName :  cost . liftGasPCM                                  Value :  19.35
     VarName :  cost . solarCost                                   Value :  0.00000000000000000
     VarName :  vehicle . beam . properties                        Value :  73.1 e9 ,0.33 ,2680....
     VarName :  vehicle . skin . properties                        Value :  112.0 e9 ,0.36 ,1440...
     VarName :  analysis . loadFile                                Value :  ../BUOYANT_FORCE/...
     VarName :  vehicle . gondola . bounds                         Value :   0.8
     VarName :  vehicle . buoyancy . HeOffsetMassFraction          Value : 20.0000000000000000
     VarName :  vehicle . mass . propulsion . engineMass           Value :  0.00000000000000000
     VarName :  vehicle . mass . propulsion . totalEngineMass      Value :  0.00000000000000000
     VarName :  solar . electricPower                              Value :  0.00000000000000000
     VarName :  vehicle . mass . solarPanelMass                    Value :  0.00000000000000000
     VarName :  vehicle . propulsion . propPower                   Value : 667200.7186740000...
     VarName :  vehicle . mass . propulsion . propFuelMass         Value : 4390.548233999999...
     VarName :  vehicle . propulsion . dragRefArea                 Value : 1430.010027999999...
     VarName :  vehicle . buoyancy . volHeAlt                      Value : 164663.8474970000...
     VarName :  vehicle . buoyancy . volHeSTP                      Value : 144502.7823459999...
     VarName :  vehicle . buoyancy . volHeLand                     Value : 138824.2595380000...
     VarName :  vehicle . skin . area                              Value : 23915.51882599999...
     VarName :  analysis . datFilePath                             Value :  ../ GEOMETRY
     VarName :  vehicle . skin . goreMaxWidth                      Value :  1.27
     VarName :  vehicle . skin . goreOverlap                       Value :  0.01
     VarName :  vehicle . skin . goreLength                        Value : 248.6151069999999947
     VarName :  vehicle . skin . numGores                          Value :    107
     VarName :  vehicle . skin . goreWidth                         Value : 1.2478290000000001
     VarName :  vehicle . buoyancy . envelope . volume             Value : 206065.2054289999...
     VarName :  vehicle . internalEnvironments . rhoIn             Value : 0.3291210000000000
     VarName :  environment . rhoOut                               Value : 1.0574140000000001
     VarName :  vehicle . buoyancy . netLift                       Value : 1472241.635999999...
     VarName :  vehicle . buoyancy . netLiftLand                   Value : 1417249.851996999...
     VarName :  vehicle . buoyancy . envelope . lift               Value : 0.0000000000000000
     VarName :  vehicle . buoyancy . liftPosition                  Value : 0.0000000000000000
     VarName :  vehicle . mass . requiredBalast                    Value : 136959.1588789999...
     VarName :  vehicle . mass . dryMass                           Value : 7510.769765000000...
     VarName :  vehicle . mass . skinMass                          Value : 2123.696399999999...
     VarName :  vehicle . mass . HeEnvMass                         Value : 2123.696399999999...
     VarName :  vehicle . mass . insulationMass                    Value : -0.0030350000000000
     VarName :  vehicle . mass . structMass                        Value : 3256.841798999999...
     VarName :  vehicle . mass . node . totalMass                  Value : 6.5400000000000000
     VarName :  vehicle . mass . burnerMass                        Value : 0.0000000000000000
     VarName :  vehicle . mass . finMass                           Value : 0.0000000000000000
     VarName :  vehicle . mass . payloadMass                       Value : 115145.0960270000...
     VarName :  environment . tempOut                              Value : 278.2440000000000282
     VarName :  vehicle . internalEnvironments . tempIn            Value : 278.2440000000000282
     VarName :  vehicle . internalEnvironments . tempInF           Value : 41.1691999999999965
     VarName :  environment . pAmbient                             Value : 84440.88000000000...
     VarName :  vehicle . mass . buoyancyFuelMass                  Value : 0.0000000000000000
     VarName :  vehicle . mass . buoyancyFuelTankMass              Value : 0.0000000000000000
     VarName :  vehicle . passengers                               Value :    959
     VarName :  vehicle . beam . MinLength                         Value : 0.5559360000000000
     VarName :  vehicle . beam . MaxLength                         Value : 19.5814860000000017
     VarName :  vehicle . beam . totalMatLength                    Value : 7578.644389000000...
     VarName :  vehicle . beam . quantity                          Value :    648
     VarName :  vehicle . node . quantity                          Value :    218
     VarName :  vehicle . htFlux                                   Value : 0.0000000000000000
     VarName :  vehicle . performance . energyPerTonKm             Value : 535.5877610000000004
     VarName :  vehicle . performance . energyPerKm                Value : 5.1050610000000001
     VarName :  cost . beamRivetCost                               Value :  0.00
     VarName :  cost . totalRivetCost                              Value : 0.0000000000000000
     VarName :  vehicle . beam . totalBeamSupports                 Value :    17481
     VarName :  time . beamFab                                     Value : 32.7250000000000014
     VarName :  time . nodeFab                                     Value : 144.2182500000000118
     VarName :  time . skinFab                                     Value : 182.8874089999999910
     VarName :  time . envelopeFab                                 Value : 182.8874089999999910
     VarName :  time . integration                                 Value : 0.0000000000000000
     VarName :  time . totalFabrication                            Value :    0.0
     VarName :  cost . laborPerHour                                Value :  0.0
     VarName :  cost . labor                                       Value : 0.0000000000000000
     VarName :  cost . mfgCostPerWatt                              Value : -1.0000000000000000
     VarName :  time . total                                       Value : 542.7180690000000141
     VarName :  solar . sizedPower                                 Value :    0.0
     VarName :  vehicle . mass . batteryPack                       Value : 163.1999999999999886
     VarName :  cost . batteryPack                                 Value : 29760.00000000000...
     VarName :  vehicle . battery . cellQuantity                   Value :    4800


       #####################################
       ##                                 ##
       ##       COEUS  TERMINATED         ##
       ##                                 ##
     ########################################
```

The primary outputs of the run are stored in one of two files. Both files store the same information; however, one is in a similar format as the variable file and one is in a more readable, hierarchical structure. Below is the the hierarchical version of the output file showing the inputs and final outputs. This format is useful in finding specific variables by system or subsystem, so long as the project has been setup appropriately.

```
analysis
    name                              A001P
    strPath                           /home/ian/Codes/structTool
    refinements                       1
    smoothRefinements                 0
    loadFile                          ../BUOYANT_FORCE/load.bdf
    datFilePath                       ../GEOMETRY
vehicle
    geometry
        geomType                      1
        length                        239.0
        diameterY                     42.5000000000000000
        diameterZ                     42.5000000000000000
        AspectRatio                   -5.
        maxDiamLocation               0.45
        aftRadiusPct                  0.1
        foreBodyPow                   2.5
        crossPow                      2.0
    beam
        area                          0.00001
        properties                    73.1e9,0.33,2680.0,1.0e8,6.35e-3,0.0002032
        MinLength                     0.5559360000000000
        MaxLength                     19.58148600000000017
        totalMatLength                7578.6443890000000465
        quantity                      648
        totalBeamSupports             17481
    insulation
        Thickness                     0.0000001
        ThermalCond                   0.0169
        Density                       0.0000001
    skin
        areaDensity                   0.074
        envDensity                    0.074
        properties                    112.0e9,0.36,1440.0,2.0e8,1.5875e-3
        area                          23915.5188259999995353
        goreMaxWidth                  1.27
        goreOverlap                   0.01
        goreLength                    248.61510699999999947
        numGores                      107
        goreWidth                     1.2478290000000001
    buoyancy
        combustEfficiency             0.85
        heatOfCombustion              46.00
        liftGas                       hybrid
        HeOffsetMassFraction          20.0000000000000000
        volHeAlt                      164663.8474970000097528
        volHeSTP                      144502.7823459999926854
        volHeLand                     138824.2595380000129808
        envelope
            volume                    206065.2054289999941830
            lift                      0.0000000000000000
        netLift                             1472241.6359999999403954
        netLiftLand                         1417249.8519969999324530
        liftPosition                        0.0000000000000000
    propulsion
        dragCoef                            0.015
        bsfc                                0.45
        numberOfEngines                     0
        propPower                           667200.7186740000033751
        dragRefArea                         1430.0100279999999202
    finMassFraction                         0.0
    mass
        node
            unitMass                        0.025
            totalMass                       6.5400000000000000
        propulsion
            engineMass                      0.0000000000000000
            totalEngineMass                 0.0000000000000000
            propFuelMass                    4390.5482339999998658
        solarPanelMass                      0.0000000000000000
        requiredBalast                      136959.1588789999950677
        dryMass                             7510.7697650000000067
        skinMass                            2123.6963999999998123
        HeEnvMass                           2123.6963999999998123
        insulationMass                      -0.0030350000000000
        structMass                          3256.8417989999998099
        burnerMass                          0.0000000000000000
        finMass                             0.0000000000000000
        payloadMass                         115145.0960270000068704
        buoyancyFuelMass                    0.0000000000000000
        buoyancyFuelTankMass                0.0000000000000000
        batteryPack                         163.1999999999999886
    gondola
        bounds                              0.8
    internalEnvironments
        rhoIn                               0.3291210000000000
        tempIn                              278.2440000000000282
        tempInF                             41.1691999999999965
    passengers                              959
    node
        quantity                            218
    htFlux                                  0.0000000000000000
    performance
        energyPerTonKm                      535.5877610000000004
        energyPerKm                         5.1050610000000001
    battery
        cellQuantity                        4800
operation
    altitude
        landing                             100.
        nominal                             1524.
        maximum                             1524.
    range                                   2088.0000000000000000
    designEndurance                         24.0000000000000000
    tempDelta                               0.0
    negTempDelta                            0.0
margin
    massMargin                              1.2
    htFluxMargin                            1.1
solar
    surfSolarFlux                           1000.0
    panelSpecificMass                       1.8
    solarPanelNu                            0.10
    electricPower                           0.0000000000000000
    sizedPower                              0.0
```

```
environment
    velocity                    87.0000000000000000
    rhoOut                      1.0574140000000001
    tempOut                     278.2440000000000282
    pAmbient                    84440.8800000000046566
cost
    costPerNode                 3.0
    costPerBeamMaterial         9.55
    solarCostPerWatt            2.00
    costPerSkinSqM              6.10
    costPerEnvSqM               5.05
    totalNodeCost               719.3999999999999773
    totalBeamCost               505.5464440000000081
    skinCost                    160473.0050000000046566
    envelopeCost                132850.6025000000081491
    totalCost                   3370294.9089440000243485
    costMargin                  1.1
    liftGas                     3075746.3549999999813735
    liftGasPCM                  19.35
    solarCost                   0.0000000000000000
    beamRivetCost               0.00
    totalRivetCost              0.0000000000000000
    laborPerHour                0.0
    labor                       0.0000000000000000
    mfgCostPerWatt              -1.0000000000000000
    batteryPack                 29760.0000000000000000
time
    beamFab                     32.7250000000000014
    nodeFab                     144.2182500000000118
    skinFab                     182.8874089999999910
    envelopeFab                 182.8874089999999910
    integration                 0.0000000000000000
    totalFabrication            0.0
    total                       542.7180690000000141
```

There are other outputs of the system in its presently configured state that are a result of the modules and not the architecture. Some of these include finite element models, structural loading results, fabrication procedures, cut schedules, and gore pattern files for the skin.

4.7.2 Example Sensitivity Study

The sensitivity study is useful in a design environment when the user wishes to understand how the system is affected by changes in one variable at a time. COEUS is designed to be capable of executing multiple sensitivities in a single run. However, this is not to imply that the interactions of these different variables are captured. In actuality, only a single variable is run at a time, but many cases can be batched together in one run.

In the example below it is of interest to understand the sensitivity of the nominal velocity, nominal altitude, and the vehicle's aspect ratio. This case was set up to perform sensitivities of all three variable from the baseline configuration used in the previous example. This sensitivity consisted of 9 data points ranging from chosen upper and lower bounds. Table 4.1 summarizes the sensitivity constraints for this example.

Table 4.1: Sensitivity example constraints summary

Sensitivity Variable	Number of Cases	Lower Bound	Upper Bound
operation.altitude.nominal	9	1200.0	1800.0
environment.velocity	9	50.0	100.0
vehicle.geometry.AspectRatio	9	5.0	15.0

Due to the quantity of information that is printed to screen in an analysis such as this one the results will not be shown in their entirety as with the previous example. However, the types of information are identical in nature. What is different in this type of analysis is how the results are stored. As there are multiple solutions run in this type of analysis each case is stored in a separate directory. This allows individual cases to be rerun if necessary, as well as providing access to additional information and outputs that may not necessarily be included in the variable file. Upon completion of this analysis, the working directory would contain three additional directories, one for each of the sensitivities to run. The naming of these directories is in the form sensitivity [Variable Name]; therefore, the results of the velocity sensitivity would be in a folder named sensitivity environment.velocity. Within each of the sensitivity folders would be a folder for each of the sensitivity cases and a file that contains the results of case.

```
COEUS ( COncurrent            Engineering          Utility      for          Systems )
POC : Ian . Dupzyk@gmail . com
Version    1.4.2


ANALYSIS  MODULES
      1 GEOMETRY  -->  NONE
      2 BUOYANT_FORCE --> NONE
      3 STRUCTURE  -->  NONE
      4 MASS_EST --> BUOYANT_FORCE
      5 GORE_DESIGN  -->  NONE
      6 COST  -->  NONE



MODULE_CONFIGURATION: GEOMETRY

      *
      *
      *

INITIAL  VEHICLE  CONFIGURATION:

VARIABLE  LIST

     VarName :  analysis . name                              Value :  A001P
     VarName :  analysis . strPath                           Value :  / home / ian / Codes / s ...
     VarName :  vehicle . geometry . geomType                Value :    1
     VarName :  vehicle . geometry . length                  Value :  239.0
     VarName :  vehicle . geometry . diameterY               Value :  42.5

     *
     *
     *

     VarName :  vehicle . beam . area                        Value :  0.00001
     VarName :  operation . altitude . landing               Value :    100.
     VarName :  time . total                                 Value :  0.0
     VarName :  solar . sizedPower                           Value :    0.0
     VarName :  vehicle . mass . batteryPack                 Value :  0.0
     VarName :  cost . batteryPack                           Value :    0.0
     VarName :  vehicle . battery . cellQuantity             Value :    0

<    WARNING >>> MODULE OUTPUT VARIABLE " vehicle . beam . totalBeamSupports " IS NOT OF TYPE
     check the declaration in the case . var file , double type should contain a decimal This variable will not be used
     in determining convergance of a feedback module
<    WARNING >>> MODULE OUTPUT VARIABLE " vehicle . passengers " IS NOT OF TYPE DOUBLE check
     the declaration in the case . var file , double type should contain a decimal This variable will not be used in
     determining convergance of a feedback module
<    WARNING >>> MODULE OUTPUT VARIABLE " vehicle . beam . quantity " IS NOT OF TYPE DOUBLE check the
     declaration in the case . var file , double type should contain a decimal This variable will not be used in
     determining convergance of a feedback module
<    WARNING >>> MODULE OUTPUT VARIABLE " vehicle . node . quantity " IS NOT OF TYPE DOUBLE check the
     declaration in the case . var file , double type should contain a decimal This variable will not be used in
     determining convergance of a feedback module
<    WARNING >>> MODULE OUTPUT VARIABLE " vehicle . battery . cellQuantity " IS NOT OF TYPE DO check
     the declaration in the case . var file , double type should contain a decimal This variable will not be used in
     determining convergance of a feedback module
<    WARNING >>> MODULE OUTPUT VARIABLE " vehicle . skin . numGores " IS NOT OF TYPE DOUBLE check
     the declaration in the case . var file , double type should contain a decimal This variable will not be used in
     determining convergance of a feedback module
```

```
Running   Sensitivity: environment.velocity   case   number: 1

MODULE: GEOMETRY ITERATION: 1

MODULE: BUOYANT_FORCE ITERATION: 1

MODULE: STRUCTURE   ITERATION: 1

MODULE: MASS_EST ITERATION: 1
      VARIABLE: vehicle.mass.skinMass               <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.mass.insulatio...           <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.mass.payloadMass            <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.internalEnviro...           <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.beam.MinLength              <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.beam.MaxLength              <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.beam.totalMatL...           <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.mass.dryMass                <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.mass.HeEnvMass              <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.propulsion.pro...           <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.mass.propulsio...           <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.mass.node.tota...           <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.performance.en...           <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.performance.en...           <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.mass.batteryPack            <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: cost.batteryPack                    <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: operation.designEndurance           <<< CONVERGENCE NOT ACHIEVED >>>

MODULE: BUOYANT_FORCE ITERATION: 2

<     MODULE CONVERGED IN 2 ITERATIONS >>>

MODULE: STRUCTURE ITERATION: 1

MODULE: MASS_EST ITERATION: 1
      VARIABLE: vehicle.mass.payloadMass            <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.mass.dryMass                <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.performance.en...           <<< CONVERGENCE NOT ACHIEVED >>>
      VARIABLE: vehicle.mass.requiredB...           <<< CONVERGENCE NOT ACHIEVED >>>

MODULE: BUOYANT_FORCE ITERATION: 2

<     MODULE CONVERGED IN 2 ITERATIONS >>>

MODULE: STRUCTURE ITERATION: 1

MODULE: MASS_EST       ITERATION: 1
      VARIABLE: vehicle.mass.payloadMass            <<< CONVERGENCE NOT
      VARIABLE: vehicle.mass.dryMass                ACHIEVED >>> <<< CONVERGENCE
                                                    NOT ACHIEVED >>>

      *
      *
      *

Running   Sensitivity: environment.velocity   case   number: 3

      *
      *
      *

Running   Sensitivity: operation.altitude.nominal        case   number: 1

      *
      *
      *
```

```
MODULE : BUOYANT_FORCE ITERATION : 1

MODULE : STRUCTURE  ITERATION : 1

MODULE : MASS_EST ITERATION : 1
     VARIABLE : vehicle . mass . skinMass          <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . insulatio ...     <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . payloadMass       <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . beam . MinLength         <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . beam . MaxLength         <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . dryMass           <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . HeEnvMass         <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . propulsion . pro ...     <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . propulsio ...     <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . node . tota ...   <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . performance . en ...     <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . requiredB ...     <<< CONVERGENCE NOT ACHIEVED >>>

MODULE : BUOYANT_FORCE ITERATION : 2

<    MODULE CONVERGED IN 2 ITERATIONS >>>

MODULE : STRUCTURE ITERATION : 1

MODULE : MASS_EST ITERATION : 1
     VARIABLE : vehicle . mass . payloadMass       <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . performance . en ...     <<< CONVERGENCE NOT ACHIEVED >>>
     VARIABLE : vehicle . mass . requiredB ...     <<< CONVERGENCE NOT ACHIEVED >>>

MODULE : BUOYANT_FORCE ITERATION : 2

<    MODULE CONVERGED IN 2 ITERATIONS >>>

MODULE : STRUCTURE ITERATION : 1

MODULE : MASS_EST ITERATION : 1

MODULE : GORE_DESIGN ITERATION : 1

MODULE : COST ITERATION : 1


#####################################
##                                 ##
##          COEUS TERMINATED       ##
##                                 ##
#####################################
```

To demonstrate the results of the sensitivities, plots of payload mass versus velocity, altitude, and aspect ratio are presented in Figures 4.5, 4.6, and 4.7, respectively.

The result of this analysis indicate the following. For Velocity, there is an inverse relationship with payload mass. As the velocity increases the payload mass that can be carried is decreased. This makes sense as the quantity of fuel required to travel at higher speeds is greater. This is due to the fact that, for a given configuration, the

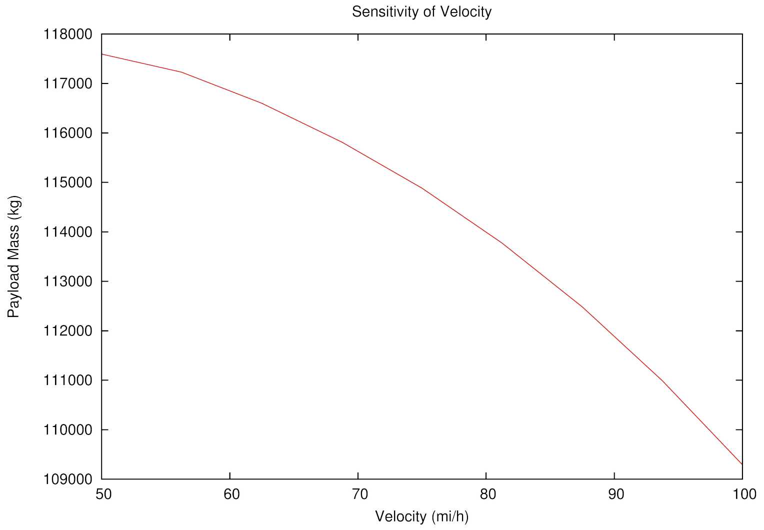


Figure 4.5: Sensitivity of payload mass to nominal velocity (mi/h)

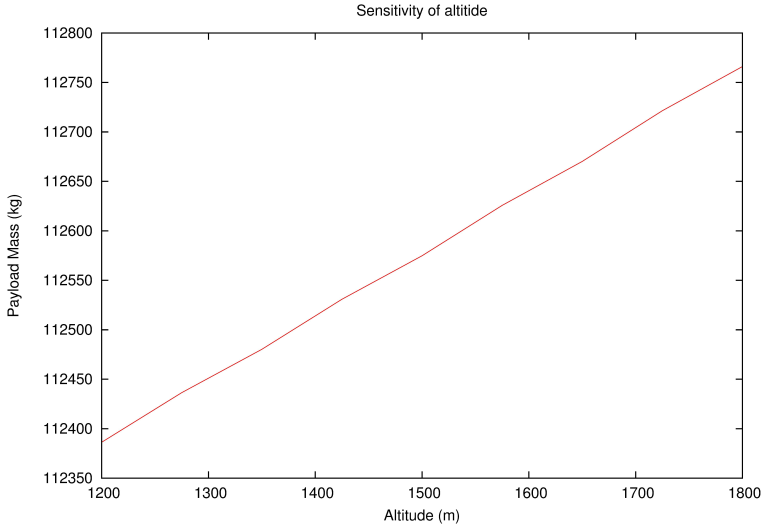


Figure 4.6: Sensitivity of payload mass to nominal altitude (m)

drag force on the system is proportionate to the square of the velocity.

The sensitivity of payload mass to the altitude has some interesting results. As the vehicle ascends higher there is more payload capacity. However, looking at the scale the amount of extra lift is rather low, having a change in payload of only 400 kg. This change in payload appears to contrast what one would think. However, by

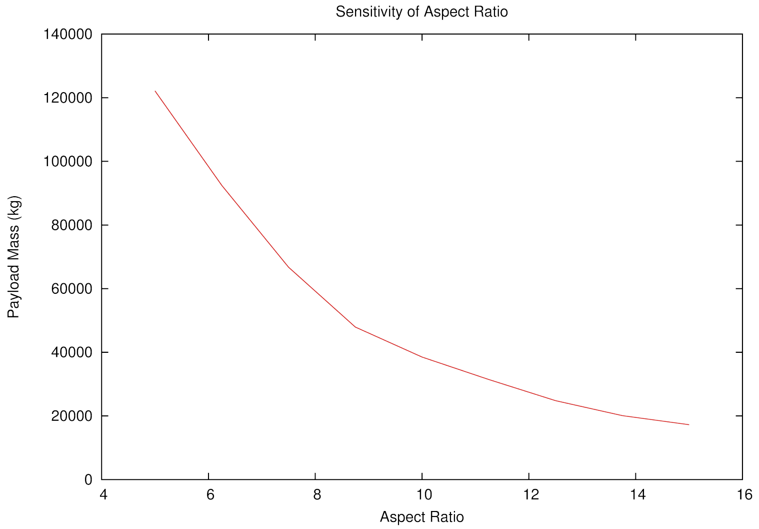


Figure 4.7: Sensitivity of payload mass to vehicle aspect ratio

looking at the whole vehicle it can be determined that this extra payload mass is actually due to less fuel required at that altitude due to decreased drag. The plot of fuel mass is shown in Figure 4.8.

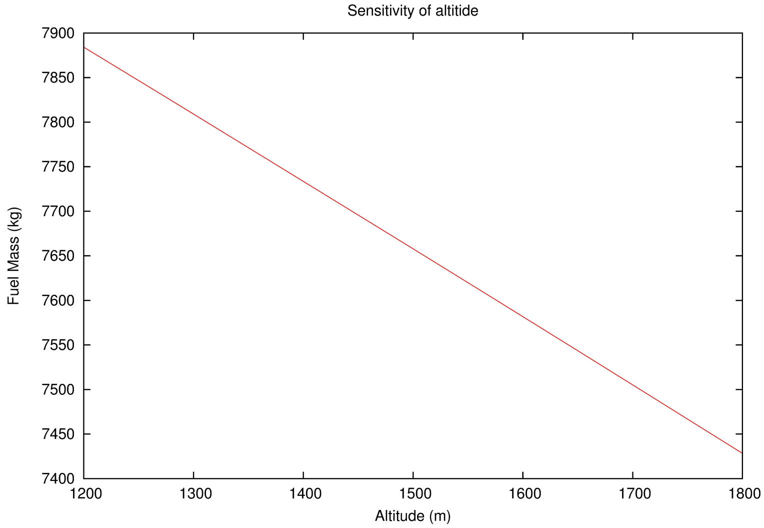


Figure 4.8: Sensitivity of propulsion fuel mass to altitude

When used in a system analysis, having the ability to run multiple sensitivity cases and look at what affects each of those variables has on the entire system can be enormously valuable to a project.

4.7.3 Example Trade Study

The following shows an example of a trade study analysis. This differs from the sensitivity study in that, a sensitivity study varies a single variable at a time. A trade study in COEUS, on the other hand, will run every combination of variable values the user specifies. While a sensitivity gives the user insight into the primary effect of a variable on the system, the trade study analysis will give the user additional insight into interaction effects.

The user inputs for the trade study are similar to that of the sensitivity study except that each value the user would like to run for a give variable is specified. Unlike the sensitivity study where the upper and lower bounds are specified along with the number of cases to run. This allows the user to cluster cases in a specific areas while still running a broad range of cases. Furthermore, clustering is important due to the large quantity of cases. Since the case list is a combination of all chosen values, the solution sets can get very large, very fast. In the example shown here, there are three variables and 9 values each resulting in 729 cases. The values in the run matrix are listed in Table 4.2. These values were chosen such that they would coincide with the values run in the sensitivity analysis for easier comparison. One can see how running additional variables at a time quickly becomes a very large problem. Even with the 729 cases in this run, being executed in series, the set took about 45 minutes to complete. The power of a tool such as this one would greatly be improved with parallelization. With the way COEUS is currently set up, parallelization would be fairly straight forward.

To demonstrate the effects of variable interactions, the following plots were devel-

Table 4.2: Trade study example run matrix

operation.altitude.nominal	environment.velocity	vehicle.geometry.AspectRatio
1200.00	50.00	5.00
1275.00	56.25	6.25
1350.00	62.50	7.50
1425.00	68.75	8.75
1500.00	75.00	10.00
1575.00	81.25	11.25
1650.00	87.50	12.50
1725.00	93.75	13.75
1800.00	100.00	15.00

oped to show the original sensitivity plots with the trade study plots superimposed. This gives an idea of the additional implications modifying a multivariate system can have and why it can be so valuable to have an architecture, like COEUS, to help understand those effects rapidly.

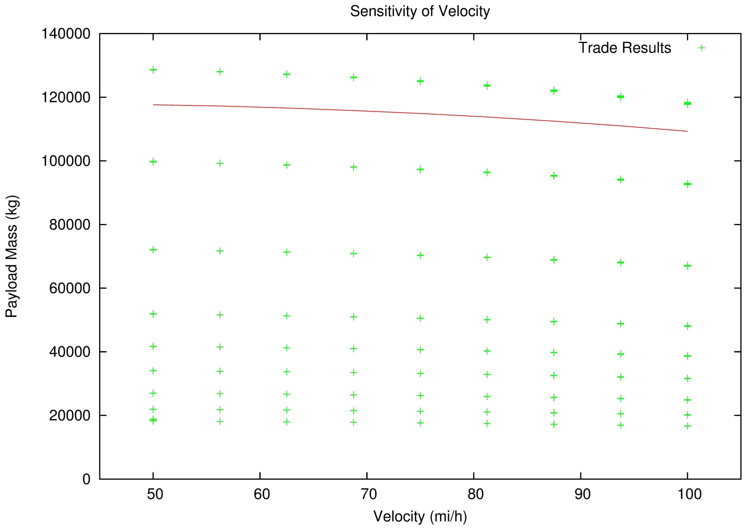


Figure 4.9: Comparison of Velocity sensitivity to Trade Study

Each of the previous plots gives some insight into the trends; however, a better way of looking at the data can be found in Figure 4.12. This plot was generated using

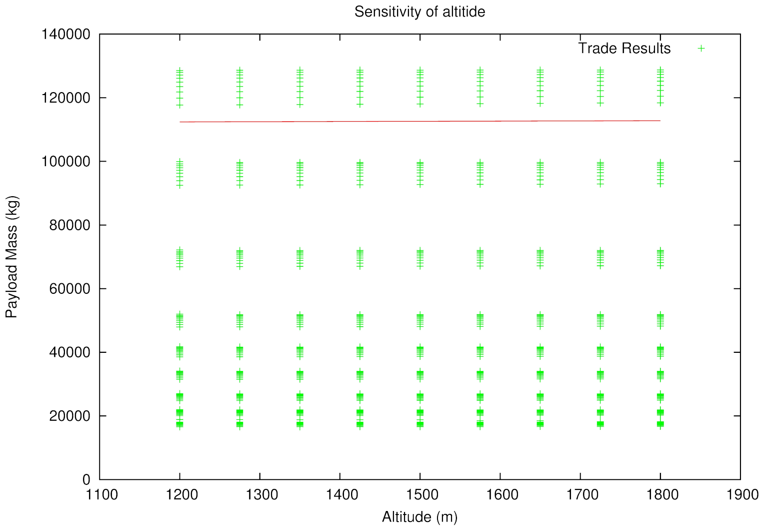


Figure 4.10: Comparison of Altitude sensitivity to Trade Study

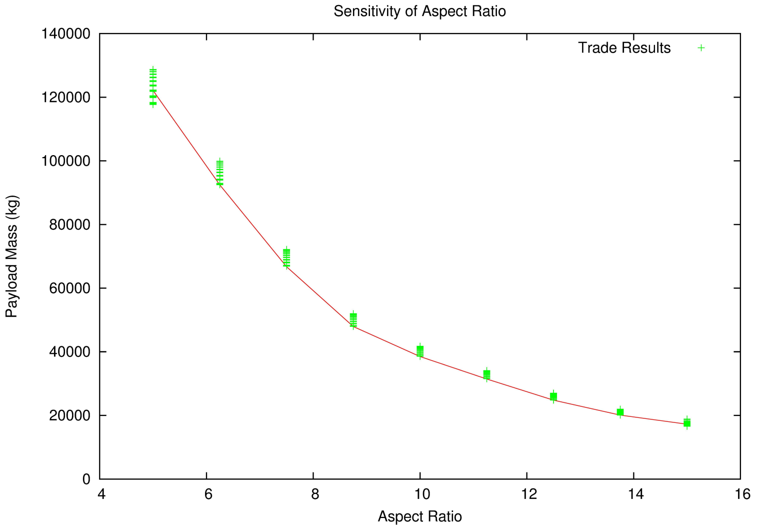


Figure 4.11: Comparison of Aspect Ratio sensitivity to Trade Study

Tableau [3] and illustrates the trends of the three variables in the study. In Figure 4.12, the plot has payload mass on the vertical axis and aspect ratio on the horizontal axis with velocity and altitude plotted parametrically. The different velocity param-eters are represented by the diameter of the circles with larger circles representing higher velocities. Altitude is represented in color, with a key on the right hand side.

From figure 4.6 it appears as though there is a reasonably strong effect of the alti-tude on the payload mass. However, in this case one can see that it is difficult to distinguish between data points of like aspect ratio and velocity indicating that in a global view there is only a small effect of that variable on the payload of the system. The aspect ratio of the vehicle appears to have the largest effect, and understandably so since the volume of the envelope (thus the lift) is substantially smaller at higher aspect ratios. Finally, the velocity of the airship has a fairly significant effect on payload mass; however, the impact diminishes as the aspect ratio increases. This is apparent by the spread in the circles from the top to the bottom. Data points at the bottom (i.e. higher aspect ratios) are more concentric than those at the top.

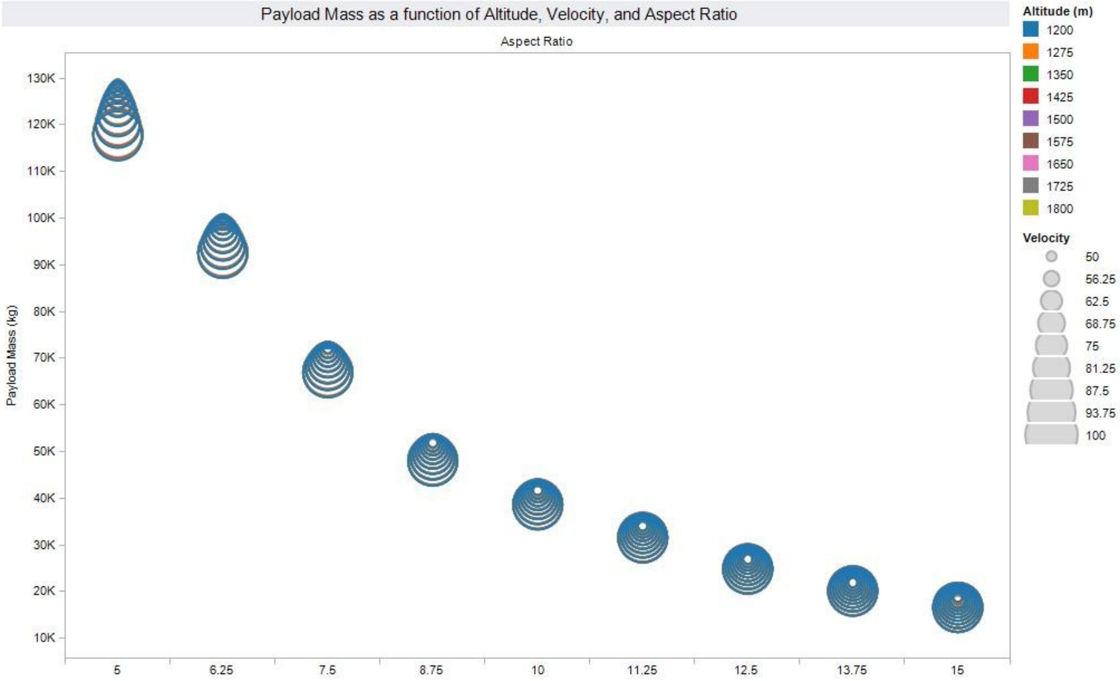


Figure 4.12: Representation of trade variables global affect on payload mass

Chapter 5

Conclusion

The work performed in this project for the degree of Master's of Science in Aerospace Engineering was the development of a multi-disciplinary design and analysis tool framework named COEUS. Its name stands for COncurrent Engineering Utility for Systems.

The goal of this framework was to develop a tool that facilitates and promotes concurrent engineering in the early design phase of a system. While the true value of COEUS would be realized at the early phases where there are a great deal of trades and system related studies that occur, there is still value in its use later in the project as simply an integration tool that executes all the modules of a system in concert. In addition to the development of COEUS, this project also covers a detailed model intended to support the design, analysis, and fabrication of a class of large rigid airships.

COEUS, itself, is not an analysis tool. It is a tool that stores information about a system and manages both the communication of vehicle information to and from the various analysis segments or modules and the execution of those modules according to a design structure matrix supplied by the user. The actual analysis is performed by user supplied modules that are tailored to the specific needs of the analysis. This fact allows COEUS to be applicable to a wide range of simulations and industries.

While COEUS is not, necessarily, industry or application specific, it is tailored to handle engineering specific problems better than other integration analysis frameworks by managing the type of study at the architecture level and not at the model level. In the design and analysis of engineering systems it is very common to run analyses, sensitivity studies, and trade studies. With traditional integration systems like Sandia's Dakota framework, the type of analysis is built into the model, i.e. the model must be set up for an analysis or sensitivity study. COEUS is different, in that the user defines a model and selects what type of analysis to perform from the analysis. The currently available analysis types offered by COEUS are simple analyses, sensitivity studies, trade studies, and simple single-variable optimization.

The examples used to demonstrate the capability of COEUS included a simple analysis, sensitivity study, and a trade study. The latter two analyses were performed on three system variables with 9 values each. These cases demonstrated the more common uses of COEUS on a representative system containing 126 variables and six analysis modules. The first of these modules creates the vehicle geometry using a parametric geometry tool developed for this purpose. The remainder of the modules perform analysis on different components or subsystems of the vehicle. One of these modules uses a high fidelity structural analysis tool and the remainder use more empirical methods demonstrating that this type of conceptual tool can incorporate higher fidelity analysis methods usually reserved for detailed analysis phases.

The current state of COEUS shows promise in conceptual and early phase system studies. There is; however, additional work that would improve the capability and performance of this tool. First and foremost would be parallelization of the analysis. As this is designed to be a concurrent engineering tool, the parallelization could be done in a number of ways. Each analysis from start to finish could be done as a separate thread or each module could be executed in parallel. The second area of work would be to improve the optimization analysis to include multi-variate optimization capability. And finally, adding capability for additional analysis types such as uncertainty analysis using Monte Carlo or Latin Hyper Cube sampling methods.

Chapter 6

Appendix A

This chapter contains all the variable names and their descriptions for all the modules in the example analysis model.

6.1 GEOMETRY Inputs and Outputs

Table 6.1: GEOMETRY Module Input Variables

Input Variable Name	Variable Description
analysis.name	The name assigned to the model. This can be the same or different from the name of the entire analysis model.
vehicle.geometry.diameterY	The maximum diameter in the Y-, or PITCH axis direction.
vehicle.geometry.diameterX	The maximum diameter in the Z-, or YAW axis direction.

vehicle.geometry.length	The length of the airship structure from nose to tail along the X direction.
analysis.refinements	The number of geodesic refinements of the initial geometry to perform
analysis.smoothingRefinements	Further geodesic refinements to result in a smoother Outer Mould Line (OML) definition. Using this option will also result in curved beams as opposed to straight, two-point members.
vehicle.geometry.foreBodyPow	The power applied to the super ellipse that governs the shape of the airship forebody from maximum diameter forward in the X-Z plane.
vehicle.geometry.crossPow	A comma delimited list of values greater than 2.0 that is distributed along the length of the vehicle from nose to tail. These define the shape in the Y-Z plane. These are the powers applied to the super ellipse equation for the Y-Z plane.
vehicle.geometry.maxDiamLocation	Location of the maximum diameter location described as a fraction of the vehicle length measured from the vehicle nose.
vehicle.geometry.aftRadPct	Radius of curvature of the vehicle tail.

vehicle.geometry.AspectRatio	Ratio of the length to diameter of the vehicle. This value is used in place of manually specifying the diameters in the X and Y directions. If this value is negative it is ignored and the current values for X and Y diameters are used. If it is positive, the X an Y diameters are computed from the length of the vehicle and this value.
vehicle.geometry.geomType	Geometry selection option. The geometry program has the option to have multiple contours for the OML of the vehicle. This is an integer flag that is used to select from the available profiles.

Table 6.2: GEOMETRY Module Output Variables

Output Variable Name	Variable Description
vehicle.skin.area	Total area of the vehicle OML.
vehicle.buoyancy.envelope.volume	Total volume of the vehicle OML
vehicle.geometry.diameterY	Diameter in the Y- or PITCH axis. Used to update the Y diameter value if the aspect ratio value is positive.
vehicle.geometry.diameterZ	Diameter in the Z- or YAW axis. Used to update the Z diameter value if the aspect ratio value is positive.

vehicle.propulsion.dragRefArea	The maximum reference area used in computation of the vehicle drag based upon drag coefficient and free-stream dynamic pressure.

6.2 BUOYANT FORCE Inputs and Outputs

Table 6.3: BUOYANT FORCE Module Input Variables

Input Variable Name	Variable Description
analysis.name	The name assigned to the model. This can be the same or different from the name of the entire analysis model.
operation.altitude.nominal	Operational altitude in meters for nominal operation
operation.altitude.maximum	Maximum altitude that will be experienced during vehicle operation in meters
operation.altitude.landing	Nominal Landing altitude in meters. This is used in the active lift calculations.
operation.tempDelta	Temperature increase for increasing vehicle lift in Celsius degrees.
operation.negTempDelta	Temperature decrease for decreasing vehicle lift in Celsius degrees.
vehicle.buoyancy.envelope.volume	Volume of the vehicle OML.
vehicle.buoyancy.liftGas	Type of lift gas used by the vehicle. Option are air, Helium and Hydrogen.

vehicle.mass.dryMass	Dry mass of the vehicle.
vehicle.buoyancy.HeOffsetMassFraction	Fraction of the dry mass of the vehicle that is to be offset by Helium when active lift is not employed.
solar.solarPanelNu	Solar panel efficiency for cases with solar power.
solar.surfSolarFlux	Solar flux constant to be used in solar power computation. Typically 1000 W is used.
solar.sizedPower	Power rating of the solar panels in Watts.

Table 6.4: BUOYANT_FORCE Module Output Variables

Output Variable Name	Variable Description
vehicle.buoyancy.netLift	The total lift of the vehicle during nominal operation. Lift force is in Newtons.
vehicle.buoyancy.liftPosition	Location of the center of lift. Measured in meters from the nose of the vehicle.
environment.pAmbient	Outside ambient temperature in pascals. Value is determined from the altitude using a standard atmosphere table.
environment.tempOut	Outside temperature in Kelvin determined from the standard atmosphere table.

vehicle.internalEnvironments.tempIn	Internal envelope temperature based on the external temperature and the temperature delta specified by the user.
vehicle.internalEnvironments.rhoIn	Internal bulk density of the vehicle. This density accounts for both the volume of helium and volume of air that may also be in the vehicle as the helium volume will not fill the entire vehicle unless the vehicle is at its maximum operational altitude.
environment.rhoOut	External density determined from the standard atmosphere table.
vehicle.buoyancy.volHeAlt	Volume of helium at nominal operating altitude.
vehicle.buoyancy.volHeSTP	Volume of helium at Standard Temperature and Pressure (STP).
solar.electricPower	Electrical power output from the system in Watts if solar panels are present.
vehicle.buoyancy.envelope.lift	Lift force of the vehicle in Newtons for nominal operation.
vehicle.buoyancy.HeOffsetMassFraction	Fraction of the dry mass of the vehicle that is to be offset by Helium when active lift is not employed.
vehicle.buoyancy.volHeLand	Volume of helium in the envelope at landing.
vehicle.buoyancy.netLiftLand	Lift of the vehicle at landing.

6.3 STRUCTURE Inputs and Outputs

Table 6.5: STRUCTURE Module Input Variables

Input Variable Name	Variable Description
analysis.name	The name assigned to the model. This can be the same or different from the name of the entire analysis model.
analysis.strPath	Path to the structure analysis program executable.
vehicle.beam.properties	A comma delimited list of Young's modulus, Poisson ratio, density, allowable stress, beam material width, beam material thickness.
vehicle.skin.properties	Similar to beam properties but not currently used.
analysis.loadFile	Name of the file from the BUOYANT_FORCE module writes the forces acting on the structure.
vehicle.buoyancy.envelope.lift	Total lift of the vehicle in Newtons.

vehicle.gondola.bounds	Limiting bounds of the gondola attachment points. Any structure node within these bounds will support the gondola. This value is listed as a fraction of the length of the vehicle. For example, a value of 0.25 would mean that from the center of lift, the gondola bounds would span one quarter of the vehicle forward and one quarter aft.
vehicle.buoyancy.liftPosition	Location of the center of lift. Measured in meters from the nose of the vehicle.
vehicle.geometry.length	Length of the vehicle.
vehicle.geometry.diameterZ	Diameter in the Z- or YAW axis. Used to update the Z diameter value if the aspect ratio value is positive.
vehicle.mass.structMass	Mass of the vehicle rigid structure.
vehicle.beam.area	Initial guess of the beam material cross section in square meters.
margin.massMargin	Margin applied to the vehicle and component masses.
vehicle.geometry.geomType	Geometry selection option. The geometry program has the option to have multiple contours for the OML of the vehicle. This is an integer flag that is used to select from the available profiles.

vehicle.mass.insulationMass	Mass of the skin insulation.
vehicle.mass.skinMass	Mass of the vehicle skin.
vehicle.mass.HeEnvMass	Mass of the helium envelope mass.

Table 6.6: STRUCTURE Module Output Variables

Output Variable Name	Variable Description
vehicle.mass.structMass	Mass of the rigid structure for the vehicle.
vehicle.beam.totalBeamSupports	Number of cross supports for all the beams in the structure. Used in determining both cost and fabrication time later.

6.4 MASS EST Inputs and Outputs

Table 6.7: MASS_EST Module Input Variables

Input Variable Name	Variable Description
analysis.name	The name assigned to the model. This can be the same or different from the name of the entire analysis model.
vehicle.skin.area	Total area of the vehicle OML.
vehicle.buoyancy.envelope.volume	Volume of the vehicle OML.
vehicle.beam.area	Nominal Starting beam cross sectional area in square meters.

vehicle.beam.properties	A comma delimited list of Young's modulus, Poisson ratio, density, allowable stress, beam material width, beam material thickness.
vehicle.insulation.ThermalCond	Thermal conductivity of the skin insulation.
vehicle.insulation.Density	Density of the skin insulation.
vehicle.skin.areaDensity	Skin mass per square meter. Used to determine the total skin mass.
vehicle.skin.envDensity	Gas envelope mass per square meter. Used to determine the total skin mass.
vehicle.buoyancy.netLift	Total lift of the vehicle at nominal operating conditions.
vehicle.buoyancy.netLiftLand	Vehicle lift at landing in Newtons.
margin.massMargin	Mass margin to apply to the system. Typically 1.0 to 1.3.
operation.designEndurance	Vehicle design endurance in hours.
vehicle.internalEnvironments.tempIn	Internal temperature of the gas inside the envelope.
environment.tempOut	External temperature based on the altitude and a standard atmosphere table.
vehicle.buoyancy.heatOfCombustion	Heat of combustion for the burner fuel. This option is for an option to use fuel burning heaters to control the vehicle lift.

vehicle.buoyancy.combustEfficiency	Efficiency of converting the chemical energy of the burner fuel to thermal energy in the lifting gas.
massPerPassenger	Passenger mass to determine the number of passengers for the resulting payload mass capability.
vehicle.insulation.Thickness	Thickness of the skin insulation in meters.
vehicle.finMassFraction	Mass fraction used to determine the mass of the fins as a function of the mass of the entire structure.
margin.htFluxMargin	Margin applied to heat rate capability of the heaters.
vehicle.buoyancy.liftGas	Type of lift gas used by the vehicle. Option are air, Helium and Hydrogen.
environment.velocity	Velocity of the vehicle at nominal operating conditions. Units are miles per hour.
vehicle.propulsion.dragRefArea	Reference area used in drag calculations. This is the projected area of the largest diameter location.
vehicle.propulsion.dragCoef	Drag coefficient of the vehicle. Currently this value is input by the user but will eventually be performed with an engineering level aerodynamics code.
environment.rhoOut	Ambient air density outside the vehicle.

vehicle.propulsion.bsfc	Brake Specific Fuel Consumption (BSFC) of the engines used for propulsion.
vehicle.propulsion.numberOfEngines	Number of engines on the vehicle
vehicle.mass.structMass	Mass of the vehicles rigid structure.
vehicle.mass.node.unitMass	Unit mass of the structure nodes.
solar.electricPower	Electric power generated by the solar panels.
solar.panelSpecificMass	Panel mass per kW.

Table 6.8: MASS_EST Module Output Variables

Output Variable Name	Variable Description
vehicle.mass.skinMass	Skin mass of the vehicle
vehicle.mass.insulationMass	Insulation mass.
vehicle.mass.payloadMass	Resulting payload mass.
vehicle.mass.buoyancyFuelMass	Mass of fuel used to keep the vehicle aloft for the specified endurance.
vehicle.passengers	Total number of passengers capable of being lifted.
vehicle.internalEnvironments.tempInF	Internal temperature in Fahrenheit.
vehicle.htFlux	Heat rate for thermal losses through the skin insulation.
vehicle.beam.MinLength	Minimum beam length in the vehicle structure.
vehicle.beam.MaxLength	Maximum beam length in the vehicle structure.

vehicle.mass.finMass	Mass of the vehicle empennage.
vehicle.beam.quantity	Total number of beams in the vehicle structure.
vehicle.beam.totalMatLength	Total length of all beams in the structure.
vehicle.mass.burnerMass	Mass of the burners for the active lift control.
vehicle.mass.buoyancyFuelTankMass	Tank mass for the burner fuel. These values are taken from existing LP tanks that are commercially available.
vehicle.mass.dryMass	Mass of the vehicle less fuels and payload.
vehicle.mass.HeEnvMass	Mass of the gas envelope.
vehicle.propulsion.propPower	Propulsive power required to maintain specified velocity.
vehicle.mass.propulsion.propFuelMass	Fuel mass of the propulsion engines based on maintaining the specified velocity for the specified endurance.
vehicle.mass.propulsion.engineMass	Single engine mass for the vehicle.
vehicle.mass.propulsion.totalEngineMass	Total engine mass for the vehicle.
vehicle.node.quantity	Total number of nodes in the structure.
vehicle.mass.node.totalMass	Total mass of all the structure nodes.
vehicle.mass.solarPanelMass	Mass of the solar panels.
vehicle.performance.energyPerTonKm	Performance metric to show the amount of energy required in fuel to transport one metric ton of payload one kilometer.

vehicle.performance.energyPerKm	Performance metric to show the amount of energy per vehicle kilometer.
vehicle.mass.requiredBalast	Ballast required at landing to keep the unladen vehicle on the ground.

6.5 GORE DESIGN Inputs and Outputs

Table 6.9: GORE_DESIGN Module Input Variables

Input Variable Name	Variable Description
analysis.name	The name assigned to the model. This can be the same or different from the name of the entire analysis model.
analysis.datFilePath	Path from the GORE working directory to the data file that provides the profile shape of the vehicle. This should be a relative path.
vehicle.skin.goreMaxWidth	Maximum width of material available to manufacture the gores.
vehicle.skin.goreOverlap	Amount of overlap of adjacent gores for gluing or sewing.
vehicle.geometry.diameterY	Maximum diameter of the vehicle. This drives the number of gores needed around the circumference of the vehicle.

Table 6.10: GORE_DESIGN Module Output Variables

Output Variable Name	Variable Description
vehicle.skin.numGores	Number of gores around the circumference of the vehicle.
vehicle.skin.goreWidth	Actual width of each gore at its widest point.
vehicle.skin.goreLength	Total length of a single gore from nose of the vehicle to tail.

6.6 COST Inputs and Outputs

Table 6.11: COST Module Input Variables

Input Variable Name	Variable Description
vehicle.node.quantity	Number of nodes in the structure.
vehicle.beam.totalMatLength	Total material length for fabrication of the beams.
vehicle.beam.properties	A comma delimited list of Young's modulus, Poisson ratio, density, allowable stress, beam material width, beam material thickness.
vehicle.skin.area	Total area of the vehicle skin.
vehicle.buoyancy.volHeSTP	Volume of helium at STP conditions.
cost.costPerNode	Cost of manufacturing each node.
cost.costPerBeamMaterial	Cost of the beam material per kilogram.
cost.costPerSkinSqM	Cost of the skin material per square meter of material.

cost.solarCostPerWatt	Cost of the solar panels per Watt.
cost.costPerEnvSqM	Cost of the gas envelope material per square meter of material.
cost.liftGasPCM	Cost of the lifting gas per cubic meter.
cost.costMargin	Margin multiplier on the system cost.
solar.electricPower	Electric power capability of the solar panels.
cost.beamRivetCost	Cost per rivet for beam fabrication.
vehicle.beam.quantity	Total number of beams.
vehicle.node.quantity	Total number of nodes.
vehicle.beam.totalBeamSupports	Total number of beam supports in the entire structure.
cost.laborPerHour	Total cost of labor per hour.
vehicle.skin.numGores	Number of gores in the vehicle skin.
vehicle.skin.goreLength	Length of a single gore.

Table 6.12: COST Module Output Variables

Output Variable Name	Variable Description
cost.totalNodeCost	Total cost for all nodes in the structure.
cost.totalBeamCost	Total cost for all beams in the structure.
cost.skinCost	Cost for the vehicle skin.
cost.envelopeCost	Cost for the gas envelope.
cost.liftGas	Cost for the lifting gas.
cost.solarCost	Cost for the solar panels.
cost.totalRivetCost	Cost for the beam rivets.

cost.labor	Labor cost for fabrication of components
cost.totalCost	Total cost of the vehicle.
time.beamFab	Time to fabricate and assemble all the beams.
time.nodeFab	Time to fabricate and assemble all the nodes.
time.skinFab	Time to assemble the skin.
time.envelopeFab	Time to assemble the gas envelope
time.integration	Time to integrate all the components into the assembled vehicle.
time.total	Total time to to complete the vehicle.
cost.mfgCostPerWatt	Cost of manufacturing the system per Watt of solar power.

Bibliography

[1] David M. Anderson. Design for Manufacturability and Concurrent Engineering. CIM Press, 2010.

[2] T. R. Browning. Applying the design structure matrix to system decomposition and integration problems: A review and new directions. IEEE Transactions on Engineering Management, 48(3):292{306, 2001.

[3] P Hanrahan. Visual analysis for everyone. http://www.tableausoftware.com/whitepapers/visual-analysis-everyone, 2012.

[4] Pheonix Integration. Model center. http://www.phoenix-int.com/software/phx_modelcenter.php, 2008.

[5] Sandia National Laboratories. http://dakota.sandia.gov/software.html.

[6] Lucy C. Morse and Danial L. Babcock. Managing Engineering and Technology. Prentice Hall, 2010.

[7] Unknown. Waterfall vs. agile. http://agileintro.wordpress.com/2008/01/04/waterfall-vs-agile-methodology/, January 2008.

[8] James P. Womack, Danial T. Jones, and Danial Roos. The Machine That Changed The World. Free Press, 1990.