Preliminary Design of an Autonomous Underwater Vehicle using a Multiple-Objective Genetic Optimizer

Matthew A. Martz

A thesis submitted to the Faculty of

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE

in

Ocean Engineering

Dr. Wayne Neu, Chairman
Dr. Alan Brown
Dr. Daniel J. Stilwell

May 27, 2008
Blacksburg, Virginia

Keywords: MDO, Genetic Algorithm, AUV, UUV, Design

Preliminary Design of an Autonomous Underwater Vehicle using a Multiple-Objective Genetic Optimizer

Matthew A. Martz

ABSTRACT

The process developed herein uses a Multiple Objective Genetic Optimization (MOGO) algorithm. The optimization is implemented in ModelCenter (MC) from Phoenix Integration. It uses a genetic algorithm that searches the design space for optimal, feasible designs by considering three Measures of Performance (MOPs): Cost, Effectiveness, and Risk. The complete synthesis model is comprised of an input module, the three primary AUV synthesis modules, a constraint module, three objective modules, and a genetic algorithm. The effectiveness rating determined by the synthesis model is based on nine attributes identified in the US Navy's UUV Master Plan and four performance-based attributes calculated by the synthesis model. To solve multi-attribute decision problems the Analytical Hierarchy Process (AHP) is used. Once the MOGO has generated a final generation of optimal, feasible designs the decision-maker(s) can choose candidate designs for further analysis. A sample AUV Synthesis was performed and five candidate AUVs were analyzed.

ACKNOWLEDGEMENTS

I would like to express my appreciation to the following:

Dr. Wayne Neu my committee chair and mentor for allowing me the opportunity to work with him. He has given me immeasurable help, advice and guidance throughout my academic career here at Virginia Tech.

Dr. Alan Brown my committee member and mentor for his all of his help and hard work.

Dr. Daniel Stilwell my committee member for the opportunity to work with him on AUVs.

Brian McCarter who was patient with me and helped me on a number of electronic component questions.

The Aerospace and Ocean Engineering Faculty and Staff for their assistance (and for being the best department on campus).

My family and friends for their guidance and support.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Traditionally the Autonomous Underwater Vehicle (AUV) design process has been largely “ad hoc” with designs governed by experience and rules of thumb. The process developed herein uses a Multiple Objective Genetic Optimization (MOGO) algorithm. This genetic algorithm searches the design space for optimal, feasible designs by considering objective attributes.

Examples of alternate optimizations methods would be the Monte Carlo, Particle Swarm Optimization (PSO), and Gradient Descent Algorithm. The Monte Carlo method relies on repeated random sampling to compute its results. Monte Carlo methods are typically used when a deterministic algorithm is not available. It is a brute force method, but it generates its models at random and is computationally intensive. Since it calculates models at random you would ideally need to go through the majority of the design space to ensure the optimal design is, in fact, optimal.

Particle Swarm Optimization bases its optimization problem in a search space and uses a swarm intelligence based algorithm to find a solution. The algorithm predicts social behavior of the swarm in the presence of objectives. PSO is another type of evolutionary algorithm with the primary difference being it uses social-psychological principles to do the optimization.

Gradient Descent Algorithm takes steps proportional to the negative of the gradient of the objective function. This requires that a single-objective deterministic algorithm is available, which is not the case with AUV Synthesis.

The use of a Genetic Algorithm (GA) was chosen for this project based on experience and resources available. The objective attributes considered for this model are: cost, risk and effectiveness. Each design is optimized to maximize effectiveness and minimize cost and risk. Genetic algorithms allow for a total systems approach to be integrated into the design process. The effectiveness classifications are defined using the US Navy UUV Master Plan (US Navy, 2004) as a baseline. The optimization results are used to create a non-dominated frontier and a baseline design is chosen for further development. A non-dominated solution, for a given problem and constraints, is a feasible solution for which no other feasible solution exists which is better in one objective attribute and at least as good in all others (Brown, 1998).

The multi-objective optimization is implemented in ModelCenter (MC) from Phoenix Integration. ModelCenter is a computer-based design integration environment that includes tools for linking design model components, visualizing the design space, performing trade studies and optimization, developing parametric models of the design space, and archiving results from multiple studies. By automating and simplifying these tasks, ModelCenter makes the

design process more efficient, saves engineering time, and reduces the error in the design process.

1.1 Motivation

A total-system approach to AUV design makes an already complex problem more complex. The goal of a total system approach is to optimize the life cycle cost-risk-effectiveness of the total AUV system. This system includes the AUV and everything outside the AUV that either affects it or is affected by it. It usually requires an iterative and interactive process that depends on an effective concurrent engineering organization to produce a true total-system result (Brown, 1998). The total-system approach illustrated above is beyond the scope of this project.

The goal of the concept design process is to identify feasible, non-dominated concepts so the decision-makers can base their selection on the objective attributes. There should be no bias for particular vehicle characteristics. Vehicle characteristics are only fundamental and intermediate parameters that lead to the calculation of the objective attributes: cost, effectiveness and risk.

Multi-Disciplinary Optimization (MDO) is essential for the design of highly integrated systems, such as AUVs, because it integrates multiple disciplines so that effective system-wide decisions can be made. The optimal system design is often something unique or non-intuitive and by using MDO it increases the confidence that the optimal design variables have been used to achieve the best design possible. Around the world MDO is increasingly being used in conceptual design problems. "For example, recently MDO has been used to reconfigure ship propulsion plants, to reduce the vehicle body weight of new cars, and to design hypersonic aircraft." (Goode, 2006)

1.2 Background

Multi-disciplinary ship and submarine synthesis models have been underdevelopment at Virginia Tech's Aerospace and Ocean Engineering Department for several years. The history of its development, additions, and modifications can be found in Brown (1998), Chen (1999), Stepanchick (2006), and Goode (2006). A submarine synthesis model used in Dr. Brown's Ship Design course (Maines et al., 2007) was customized to develop this particular model.

The submarine synthesis model builds and balances a design based on specified inputs and estimates its feasibility, effectiveness, cost and risk. The submarine synthesis model uses 14 modules to generate a submarine model. These modules are:

- Input
- Combat Systems
- Propulsion
- Hull

- Tankage
- Space
- Electric
- Resistance
- Weight
- Feasibility
- Overall Measure of Effectiveness (OMOE)
- Cost
- Risk
- MOGO

The AUV Synthesis Model has simplified some of these and operates on different assumptions. The AUV Model doesn't have the need for the submarine's Combat Systems and Tankage modules. They would be considered "payload" if included at all within the model. The AUV Hull module has combined the submarine's Space and Weight modules. The AUV Synthesis Model's module descriptions are discussed in Section 3.2.

1.3 Thesis Overview

Chapter 1 provides an introduction, motivation for the use of synthesis models, and background work related to using synthesis models for design.

Chapter 2 discusses the Multiple-Objective Genetic Optimization process in detail.

Chapter 3 shows how the AUV Synthesis Model is setup.

Chapter 4 demonstrates the setup and evaluation of an AUV synthesis.

Chapter 5 does an in-depth analysis of how the principle AUV characteristics affect the Measures of Performance.

Chapter 6 is a summary and conclusion, including possible future work.

Chapter 2 Multiple-Objective Genetic Optimization

A Multiple-Objective Genetic Optimization (MOGO) is used to search the design space to develop a set of optimal feasible AUV designs. The MOGO uses a genetic algorithm to improve a population of potential optimum designs. Initially the MOGO chooses 200 random designs from the design space. It runs each design through the synthesis model to determine feasibility, effectiveness, risk and cost. The genetic algorithm compares each design to determine their relative dominance and generates a new generation of 200 designs from the most dominant designs. Dominance is defined as the design with the best effectiveness, for a given cost and risk. The genetic algorithm is explained in more detail in Section 2.1.

The MOGO process is illustrated in Figure 1. The optimization process runs through hundreds of generations of designs and provides multiple baseline designs in a non-dominated frontier.

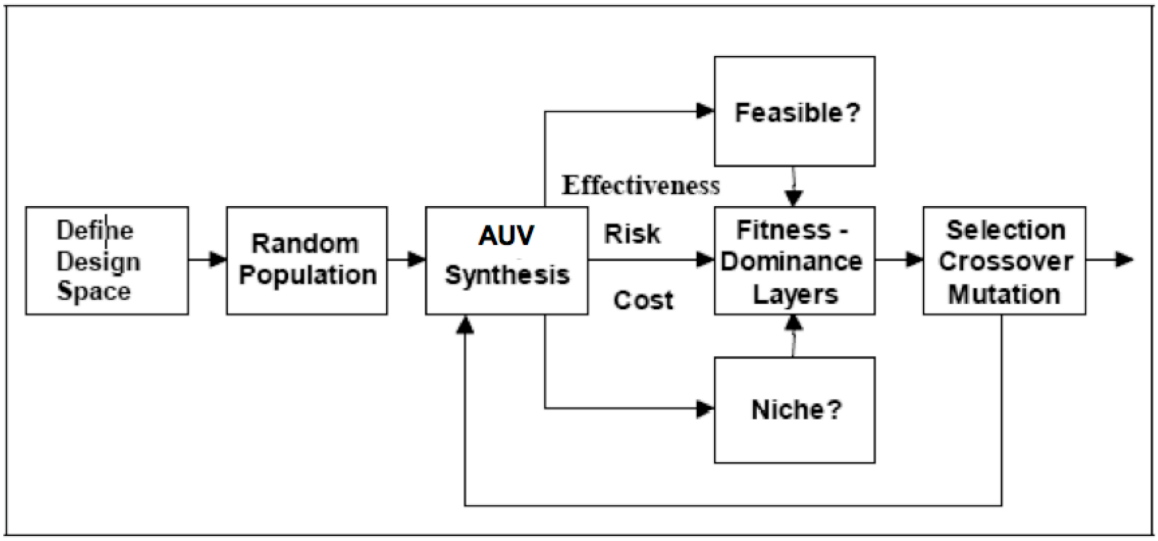


Figure 1: Multiple-Objective Genetic Optimization Process (after Brown, 1998)

2.1 Genetic Algorithms

Genetic algorithms take the adaptive processes of natural systems and apply them to the design of artificial systems. They are classified as global search heuristics, which is a branch of applied mathematics and numerical analysis that deals with the optimization of a set of functions to some criteria. Genetic algorithms use techniques inspired by evolutionary biology including inheritance, mutation, selection, and crossover to improve a population of separate designs. Genetic algorithms are most appropriate to optimizing discontinuous, disjointed functions and where no closed-form function exists.

A genetic representation and fitness function must be developed in order for the genetic algorithm to be effective. In the case of AUV design the genetic

representation is defined by Design Variables (DVs), while the AUV's environment is defined by Design Parameters (DPs). Compared to biology, DVs would be the traits or genes of the AUV while the DPs would be environmental characteristics that define if or how the AUV will survive in the environment. All of the DVs combined would be equivalent of an AUV's DNA. The fitness function provides Measures of Performance (MOPs) with which an AUV can be compared to other AUVs.

The MOPs in a less complicated system would be some type of effectiveness function, but with such a highly integrated system multiple fitness functions are used to evaluate the models. In AUV design, the fitness functions being used are Effectiveness, Risk, and Cost. The DPs define certain parameters for the fitness functions and they are also used to create feasibility constraints for the AUV. If an AUV is outside a range for a particular feasibility constraint, the AUV will not survive to produce offspring.

Once the genetic representation and fitness functions are defined the Genetic Algorithm initializes the population of models randomly. The dominance of each model is determined relative to one another, and a new generation is created. To ensure that the new population has a rich combination of genetic material two processes are performed on the population: inheritance and mutation. Inheritance is the merging of two designs with half of the variable values being swapped between the two designs. This ensures that different combinations of good gene pools are found. Mutation is the changing of only one design variable value in a design. This ensures that the populations will use the total design space. In each generation, the fitness of every individual in the population is evaluated and the best models of the generation are used to be parents for the next generation. The traits of parents are recombined to form a new generation. Some of this new generation is then randomly mutated in order to preserve genetic diversity from one generation of a population to the next. During mutation a random gene or genes may be modified. Mutation allows the genetic algorithm to avoid local minima. This new generation is then used to spawn the following generation, and so on. The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached.

After the synthesis model has run, more sophisticated tools, models and simulations can be used on selected concepts to refine the designs. Analysis results are added to a knowledge database where model trends can be observed. The results can be used to update model equations and MOP functions. Using the results to refine the modules within the genetic algorithm enables the decision makers to reevaluate and adjust earlier design decisions, and to generate better designs with a shorter development time.

2.2 Multiple Objective Optimizations

Autonomous Underwater Vehicles bring forth some interesting challenges for designers. Decision makers must choose a design that fits multiple competing criteria and satisfy a complex set of constraints. The ideal situation would have Effectiveness maximized with Risk and Cost minimized, but in practice this is not possible. A feasible solution to a multiple objective problem is efficient if no other feasible solution is at least as good for every objective and strictly better in one.

Figure 1 (Brown, 1998) illustrates this concept for a simple two-objective cost-effectiveness problem. The non-dominated, feasible solutions are represented by the heavy curve on the left. The decision-maker's preference for cost and effectiveness guides their selection of a model, but the selection should always be one on the non-dominated frontier, or Pareto front. If a design is chosen off this frontier the decision-maker would be sacrificing either cost or effectiveness for no reason. The preference of a decision-maker may be affected by the shape of the frontier and cannot be rationally determined before the synthesis is run.

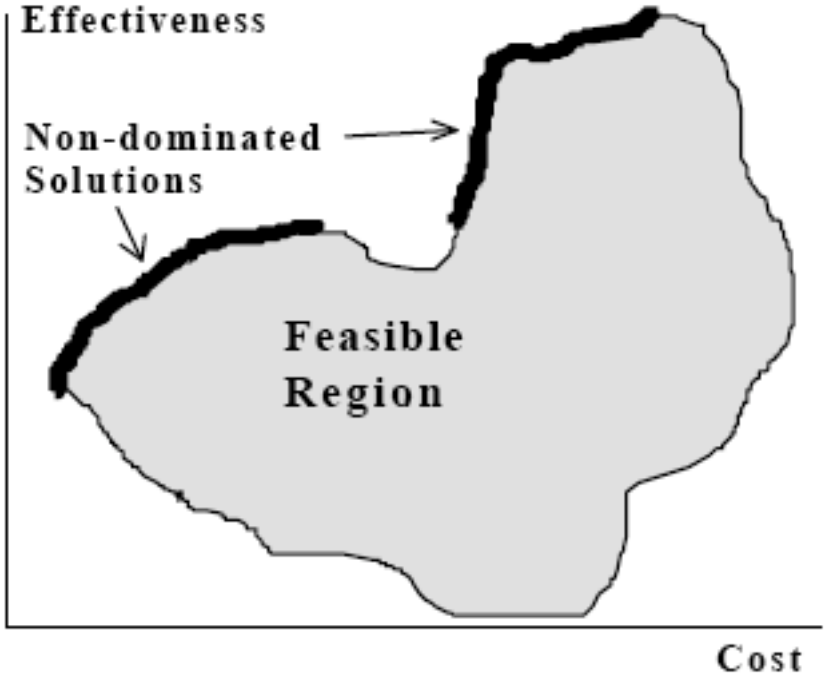


Figure 2: Two Objective Attribute Space (Brown, 1998)

When three attributes are considered, the non-dominated frontier will take the form of a surface plot. Points on this surface represent the best feasible designs. The full range of cost-risk-effectiveness possibilities can be presented to decision-makers using such a surface. Trade-off decisions can then be made using "knees in the curve" which can be seen graphically on the surface. Once specific design concepts are chosen they can be analyzed using more rigorous methods.

2.3 Analytical Hierarchy Process

Most decisions in the design process involve multiple criteria with complex relationships, and the human mind has limited capacity to consider everything at once. To solve multi-attribute decision problems the Analytical Hierarchy Process (AHP) is used. The AHP is a tool developed by Saaty (1996) that uses a hierarchical structure to abstract, decompose, organize and control the complexity of decisions that involve many attributes. It uses informed judgment of expert opinion to measure the relative value or contribution of these attributes. Pair-wise comparison is used to generate the relative weights of importance of attributes.

The first step in the construction of an AHP hierarchy is to identify critical attributes affecting the system behavior. For the Overall Measure of Effectiveness (OMOE) module in the synthesis model, the first nine attributes were identified from the UUV Master Plan (US Navy, 2004) and four additional attributes were identified by expert opinion. The 13 attributes are:
Intelligence, Surveillance and Reconnaissance
Oceanography
Mine Countermeasures
Antisubmarine Warfare
Information Operations
Payload Delivery
Time Critical Strike
Inspection/Identification
Communication/Navigation Network Node
Sprint Speed
Endurance Speed
Endurance Duration
Expandability

Next, the relative influence of each attribute on system performance and attribute values for each alternative must be estimated. Saaty's recommendation of using a nine level dominance scale for the pair-wise comparison was used in the evaluation of this model. Each attribute is compared to one another on this nine level dominance scale. The results of this are put into a 13 x 13 matrix.

The same process was used to develop the weights for the Overall Measure of Risk (OMOR) function. There are 3 types of risk associated with ship or AUV design: Performance, Schedule and Cost. A survey was used to identify the relative importance of these types of risk. A scale comparing each type or risk to each other is used to identify importance. The scale goes from 1 – 9 with 1 being type A is most important, 5 as they are equal importance, and 9 being type B is more important. When performance risk (A) was compared to schedule risk (B) it was given a 3. Performance and cost were said to be equal and schedule (A) versus cost (B) was given a 6, meaning cost was slightly more important than

schedule. These results are then put into a 3 x 3 matrix. The diagonal of the matrix are all ones. The matrix is shown below:

$$A = \begin{bmatrix} 1 & 5 & 1 \\ 1/5 & 1 & 1/3 \\ 1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 5 & 1 \\ 0.2 & 1 & 0.3333 \\ 1 & 3 & 1 \end{bmatrix} \quad (1)$$

To normalize the weights, each column is summed and then divided by the corresponding sum, which gives a new matrix of:

$$\overline{A} = \begin{bmatrix} 0.4545 & 0.5556 & 0.4286 \\ 0.0909 & 0.1111 & 0.1428 \\ 0.4545 & 0.3333 & 0.4286 \end{bmatrix} \quad (2)$$

The next step is to compute the average values of each row and use these as the weights. For this calculation the weights would be:

$$W = \begin{bmatrix} 0.4796 & 0.1149 & 0.4055 \end{bmatrix} \quad (3)$$

The sum of the weights should be 1. The code used to generate the Objective and Risk weights are listed in Appendix A and B.

Chapter 3 AUV Synthesis Model

3.1 ModelCenter

ModelCenter version 7.1 is a commercial process integration and design optimization software package developed by Phoenix Integration, Inc. used to build and execute complex engineering models. ModelCenter (MC) provides a visual environment in which design processes can be assembled as a series of linked applications with a single interface in order to easily perform multi-disciplinary analysis. MC facilitates the communication of data from one application to the next, producing an automated multi-disciplinary design environment. More information about MC, and the Darwin Optimization plug-in discussed below, can be found at http://www.phoenix-int.com/. ModelCenter was chosen for this project based on availability and experience. ModelCenter is used in the AOE Department for senior design courses.

3.1.1 Darwin Optimization Plug-In

MOGO is performed in MC using the Darwin Optimization plug-in, version 1.2.1. The Darwin Optimization plug-in is a Genetic Algorithm-based optimization technique designed to solve engineering optimization problems with multiple objectives and any number of constraints. Darwin is also capable of handling both discrete and continuous Design Variables. Each Design Variable and criterion is specified using the Darwin Optimization plug-in graphical user interface (GUI) as shown in Figure 3.

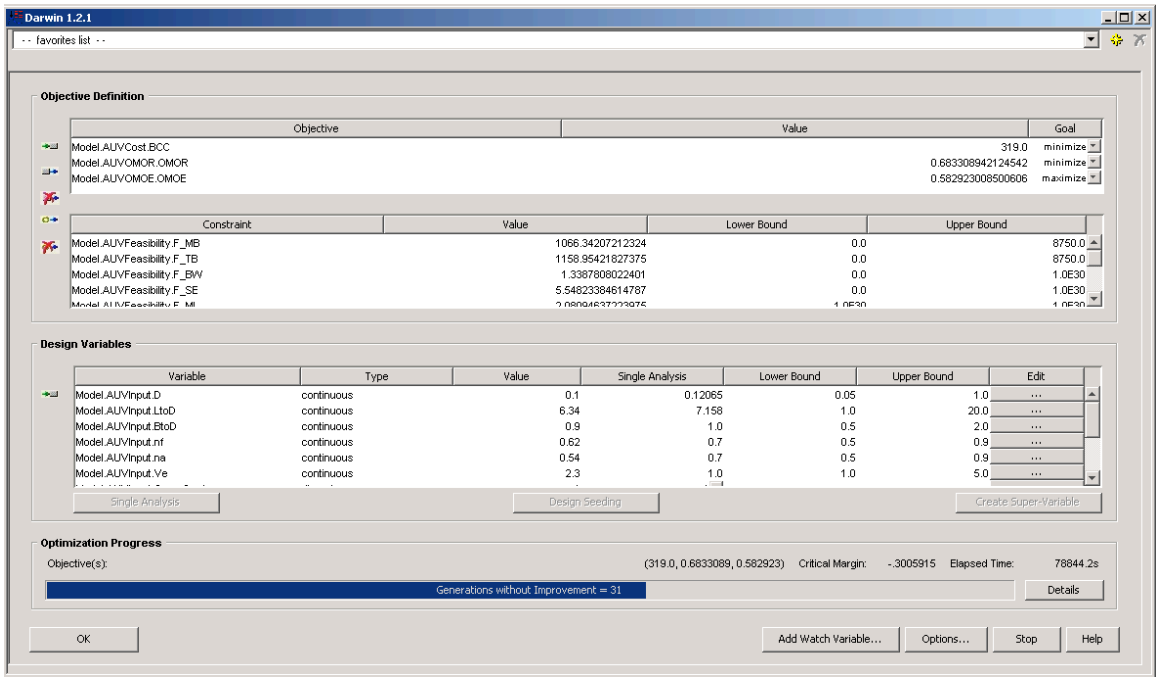


Figure 3: Darwin Optimization GUI

The user interface allows the user to modify various genetic algorithm and output parameters. One particular parameter of interest is "Selection Scheme." "Selection Scheme" determines which designs from the parent and child populations will be selected to make up the next generation of designs.

The Darwin Optimization plug-in utilizes two different selection schemes: Elitist and Multiple Elitist. The Elitist selection scheme replaces the worst design of the child population with the best design from the parent population and is typically used for optimization problems with a single global optimum that is not surrounded by a large number of local optima.

In the Multiple Elitist selection scheme, the parent and child populations are combined into one population and then ranked according to fitness. This scheme is more effective for optimization problems where the design space has multiple local optimum points and thus is used in this work.

Two other GA parameters of interest are "Population Size" and "Preserved Designs." "Population Size" refers to the number of designs in each population, while "Preserved Designs" refers to the number of designs preserved for the next population.

3.2 Modules

The optimization process for the complete model setup is displayed in Figure 4. The complete model is comprised of an input module, the primary AUV synthesis model modules, a constraint module, three objective modules, and a genetic algorithm (GA).

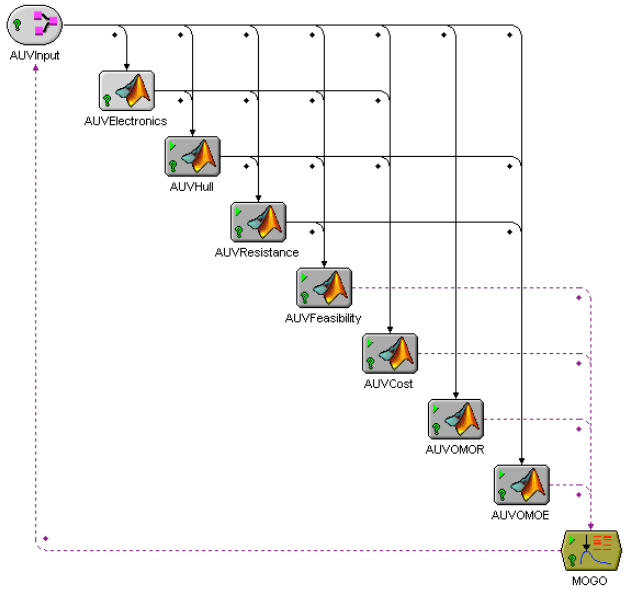


Figure 4: AUV Synthesis Model in ModelCenter

The primary AUV synthesis model is comprised of 3 modules, each focusing on one specific discipline. The three primary modules are the Electronics, Hull, and Resistance modules. Modules can easily be added or removed in the MC environment, allowing the model to more accurately represent the mission definition. All modules in this model are coded in MATLAB, but MC can also use Excel documents and FORTRAN code with the appropriate file wrappers, or one can write modules directly in MC using a BASIC-like language. The MATLAB code for each module is listed in Appendix C – I.

3.2.1 Input

The first module serves one main purpose: to collect the values of the input variables before providing them to specific modules. The values within this module are the Design Variables (DVs) and Design Parameters (DPs). The MOGO module output connects as an input to this module, allowing the optimizer to adjust input variables for different designs.

Design Parameters and Design Variables define the environment the AUV operates in and the characteristics of the AUV (respectively). Table 1: Design Parameter is a list of the Design Parameters and Table 2: Design Variable is a list of the Design Variable.

Table 1: Design Parameters used in AUV Synthesis Model

DP Name	Description
PC	Propulsive Coefficient
Eta	Motor Efficiency
Vsmin	Minimum Sprint Speed (knots)
Vsgoal	Goal Sprint Speed (knots)
Vemin	Minimum Endurance Speed (knots)
Vegoal	Goal Endurance Speed (knots)
MinDuration	Minimum AUV Duration at Endurance Speed (hours)
GoalDuration	Goal Endurance Duration (hours)
MinBallast	Minimum Ballast Mass (kg)
GoalBallast	Goal Ballast Mass (kg)
PR	Performance Risk Weight
SR	Schedule Risk Weight
CR	Cost Risk Weight
EW1	Effectiveness Weight (ISR)
EW2	Effectiveness Weight (Oceanography)
EW3	Effectiveness Weight (CN3)
EW4	Effectiveness Weight (Mine Countermeasures)
EW5	Effectiveness Weight (Anti-Submarine Warfare)
EW6	Effectiveness Weight (Inspection/Identification)
EW7	Effectiveness Weight (Payload Delivery)

EW8	Effectiveness Weight (Information Operations)
EW9	Effectiveness Weight (Time Critical Strike)
EW10	Effectiveness Weight (Sprint Speed)
EW11	Effectiveness Weight (Endurance Speed)
EW12	Effectiveness Weight (Endurance Duration)
EW13	Effectiveness Weight (Ballast/Expandability)

Table 2: Design Variables used in AUV Synthesis Model

DV Name	Description
D	Vehicle Diameter
LtoD	Length to Diameter Ratio
BtoD	Beam to Diameter Ratio
nf	Forward shape coefficient
na	Aft shape coefficient
Ve	Endurance Speed
CommConf	Communication Configuration
PayConf	Payload Configuration
PropConf	Propulsion Configuration
BatConf	Battery Configuration
ElecConf	Electronics Configuration
WallType	Hull Wall Thickness/Material

3.2.2 Electronics

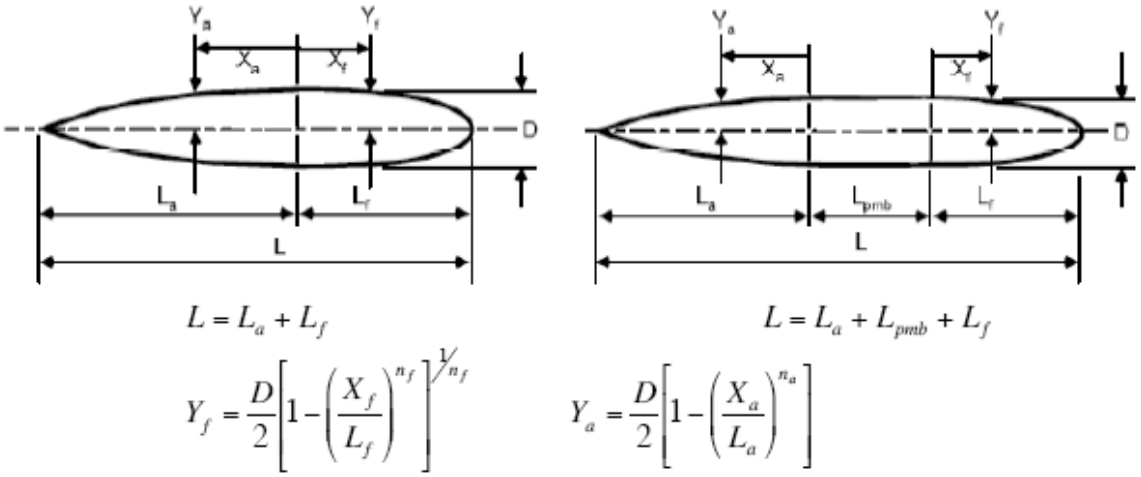
The Electronics module computes overall electrical power requirements. The input parameters are used to compute these requirements. First, non-payload power consumption is calculated by summing individual components. This value is then combined with payload to obtain the maximum functional load. Multiplying all of the vehicle component voltages with their respective current requirements and summing them together calculate the maximum functional load.

$$P_{MFL} = \sum V_i I_i \qquad (4)$$

The maximum functional load is not the average load that the components will encounter. It is used so that a conservative estimate of range and lifetime of the AUV can be considered. In a similar manner to the maximum functional load, the module also calculates the total power available. The module calculates the "port feasibility" for the Feasibility module due to a limitation of ModelCenter. "Port feasibility" assesses whether or not the Electronics and Payload packages of the AUV have enough of the right kind of data ports (more on this in Section 3.2.5).

3.2.3 Hull Geometry and Arrangement

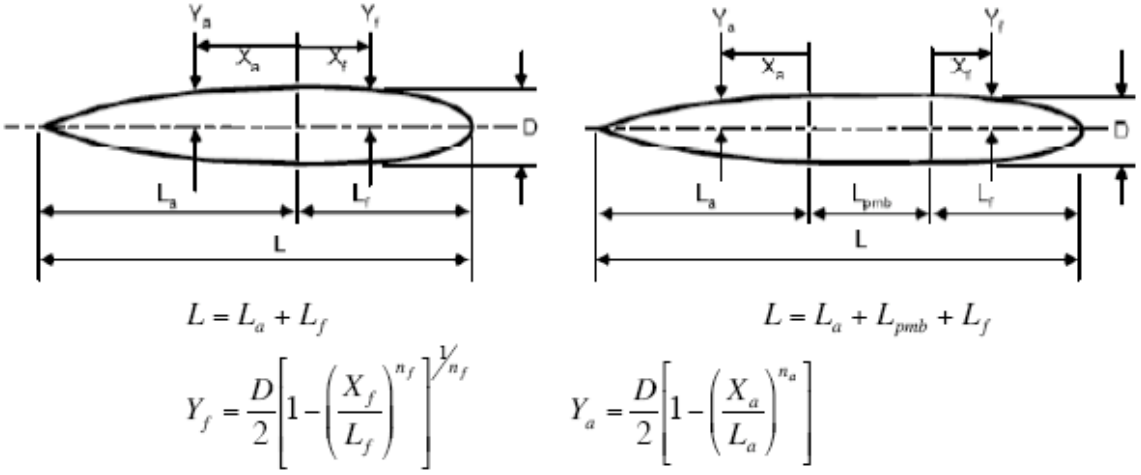
Similar to the Submarine Synthesis Model (Maines et al., 2007), an axisymmetric teardrop hullform with parallel mid-body was selected for use. The advantages of the axisymmetric hull are its producibility, low resistance, and structural efficiency. The hullform model is based on the MIT hull model (Jackson, 1992), shown in



$$L = L_a + L_f \qquad L = L_a + L_{pmb} + L_f$$

$$Y_f = \frac{D}{2}\left[1 - \left(\frac{X_f}{L_f}\right)^{n_f}\right]^{1/n_f} \qquad Y_a = \frac{D}{2}\left[1 - \left(\frac{X_a}{L_a}\right)^{n_a}\right]$$

Figure 5: Hull model; without parallel midbody (left); with parallel midbody (right). The MIT hull model used was also used for the Submarine Synthesis Model (Maines et al., 2007).

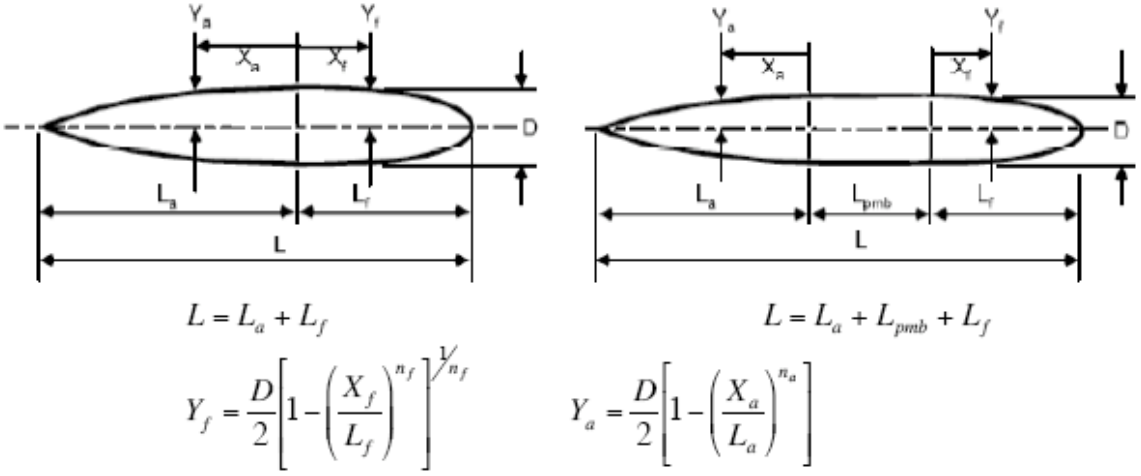


$$L = L_a + L_f \qquad L = L_a + L_{pmb} + L_f$$

$$Y_f = \frac{D}{2}\left[1 - \left(\frac{X_f}{L_f}\right)^{n_f}\right]^{1/n_f} \qquad Y_a = \frac{D}{2}\left[1 - \left(\frac{X_a}{L_a}\right)^{n_a}\right]$$

Figure 5: Hull model; without parallel midbody (left); with parallel midbody (right)

In



$$L = L_a + L_f \qquad L = L_a + L_{pmb} + L_f$$

$$Y_f = \frac{D}{2}\left[1 - \left(\frac{X_f}{L_f}\right)^{n_f}\right]^{1/n_f} \qquad Y_a = \frac{D}{2}\left[1 - \left(\frac{X_a}{L_a}\right)^{n_a}\right]$$

Figure 5, n_f and n_a are the forward and aft shape coefficients. They are used as DVs in the synthesis model and they adjust the fullness of the forward and aft sections. Also, the surface area is a function of n_f and n_a. This module assigns the forward and aft sections each to be 25% of the length overall, and the parallel mid-body being the remaining 50%.

The Hull code does a preliminary arrangement of hull items using a "box stacking" algorithm developed for this synthesis model. It begins by ensuring the components are aligned in the appropriate direction, with the longest dimension in line with the longitudinal direction and the second largest dimension aligned with the beam direction. Components are then sorted by weight, with the heaviest objects being placed first. Objects are always placed as low as possible in the AUV. If the first object placed leaves room below it for the 2nd object, the 2nd object will be placed below it. MATLAB has a function built in called "inpolygon". This function is used to ensure that two objects don't occupy the same space. Once the first section area is filled, the code steps forward in the AUV to the next free space. If object 1 is the longest object of the group, and is 30 cm long, the code steps forward 30 cm and tries to start another object at the end of it (all while continually checking to make sure nothing overlaps). This process repeats until all of the objects are placed. With all of the objects placed, the code finds the total length (or "compressed length") of the objects. The feasibility module uses this in comparison with the length of the vehicle (See Section 3.2.5).

3.2.4 Resistance

The function of this module is to perform the calculation of speed and endurance. The total resistance of the AUV is estimated based on the shape of its hull. A correlation allowance is added to the viscous resistance to calculate the total resistance. Resistance is calculated for fully submerged conditions only. The bare hull power is then calculated and shaft power is determined.

The module first incorporates hull characteristics of the AUV from the input module: Diameter, Length Overall, Beam, Volume and Surface Areas. The bare-hull skin friction coefficient, C_f, is found from the 1957 ITTC line: (Principles of Naval Architecture, 1988).

$$C_F = \frac{0.075}{\left(\log_{10} R_n - 2\right)^2} \tag{5}$$

where R_n is the Reynolds number based on the length of the AUV. The coefficient of viscous resistance for the smooth bare hull, C_{VBH}, is then found using an equation from Gillmer and Johnson (1982) modified to take into account the effects of the forward and aft shape coefficients, n_f and n_a respectively (See Section 3.2.4.1 for modification details):

$$C_{V_{BH}} = C_F\left[1 + 0.5\left(\frac{1}{LtoD}\right) + 3\left(\frac{1}{LtoD}\right)^{7-n_f-\frac{n_a}{2}}\right] \tag{6}$$

where *LtoD* is the length to diameter ratio. From this, the Effective Power (PE) and Shaft Power (PS) can be calculated:

$$PE = \frac{\rho v^3}{2}\left[\left(C_{V_{BH}} + C_A\right)S_{BH} + \sum C_{V_{AP}} S_{AP}\right] \tag{7}$$

$$PS = \frac{PE}{PC} \tag{8}$$

Where ρ is the density of water, v is the velocity, C_A is the roughness allowance for full-scale resistance estimates made without model tests, S_{BH} is the bare-hull surface area, *PC* is the propulsive coefficient, and C_{VAP} and S_{AP} are the appendage viscous coefficient and surface areas respectively. It is common to use a value of 0.0002 for C_A. C_{VAP} and S_{AP} would be found using either experimental data or computational fluid dynamics calculations.

This process is then used in reverse to calculate the maximum speed of the vehicle, given the maximum effective power output of the motor. The *PS, PE, and* R_n are all functions of the velocity. *PS* is compared to the effective power output of the motor (motor rating x efficiency of motor). The speed at which these two values match is the sprint speed of the AUV.

3.2.4.1 Resistance Code Modification

Equation 6, above, was modified by Dr Brown for his Submarine Design course. The equation was modified from the original Gillmer and Johnson (1982) equation to account for the effects of the forward and aft shape coefficients directly in the equation. The original Gillmer and Johnson equation was:

$$C_{V_{BH}} = C_F\left[1+0.5\left(\frac{1}{LtoD}\right)^{1.5}+3\left(\frac{1}{LtoD}\right)^{3}\right] \quad (9)$$

The modified Gillmer and Johnson equation was compared to the original equation and an equation used by the MIT Harry Jackson method (Jackson, 1992). The MIT Harry Jackson method uses the prismatic coefficient, C_p, of the AUV:

$$C_p = \frac{V_{env}}{\pi\left(\frac{D}{2}\right)^2 LtoD(D)} \quad (10)$$

$$C_{V_{BH}} = C_F\left[1+1.5\left(\frac{1}{LtoD}\right)^{1.5}+7\left(\frac{1}{LtoD}\right)^{3}+0.002\left(C_p-0.6\right)\right] \quad (11)$$

In equations 9 and 11 the effects of n_f and n_a on the resistance is due to their effect on the surface area. To ensure equation 6 fits with equations 9 and 11, the forward and aft shape coefficients were varied over the design space. For example, Figure 6 shows the comparison between the three methods for an AUV with a 0.5 m diameter, Length to Diameter ratio of 8, a forward shape coefficient of 0.7 and an aft coefficient of 0.5. Other forward and aft shape coefficients were tested and a similar agreement was found.

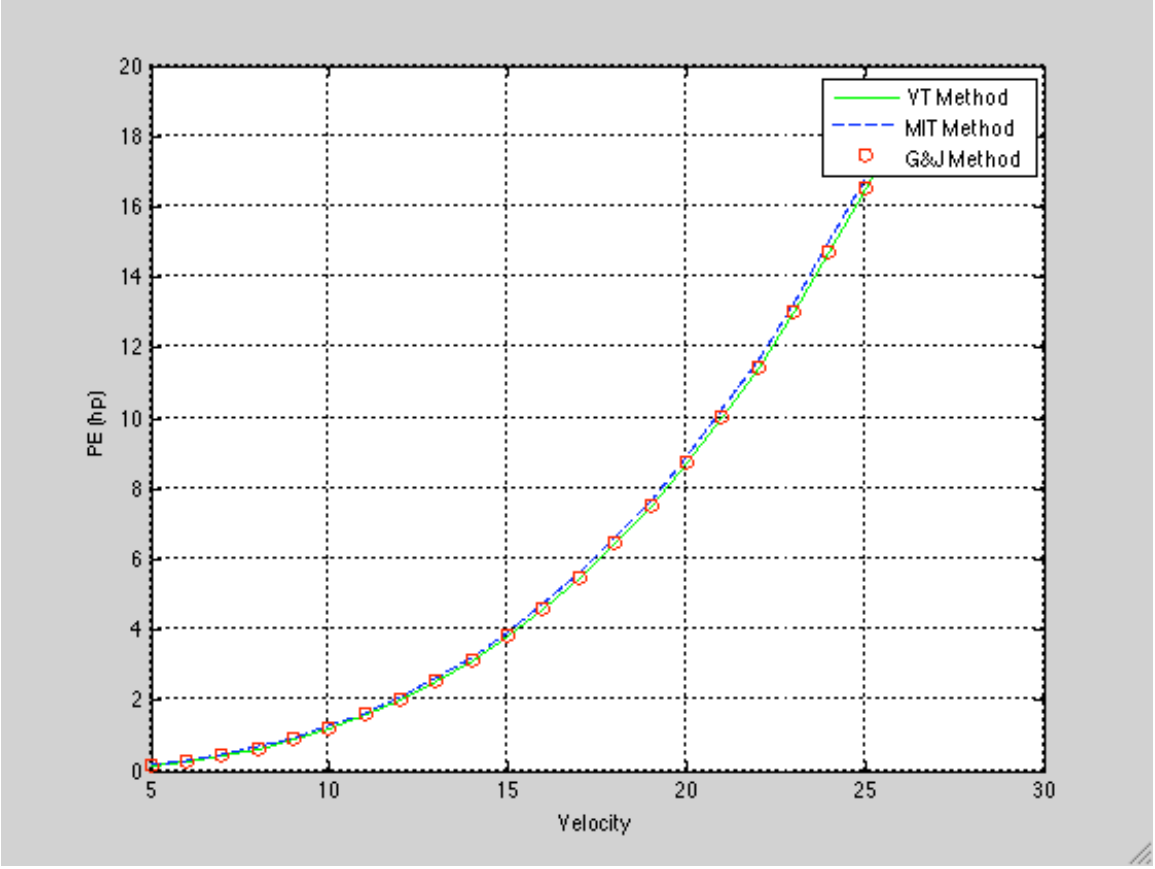


Figure 6: Resistance Comparison between the VT, MIT and Gillmer and Johnson Methods

3.2.5 Feasibility

Characteristics such as speed, range, duration, ballast, etc are all examined to determine feasibility. In order for a specific design to be feasible, all feasibility ratios must be greater than zero, and in some circumstances be below a certain threshold. Feasibility ratios are defined as:

$$FR = \frac{C - C_{\min}}{C_{\min}} \tag{12}$$

where C is the constraint in question, and C_{min} is the minimum for that constraint. The module returns values of the various comparisons, demonstrating which aspects of the design are feasible and which are not. Table 3 is a list of the constraints generated by this module. Section 4.1.4 demonstrates how to set up the constraints in ModelCenter.

Table 3: Constraints used in AUV Synthesis Model

Name	Description
F_TB	Total Ballast Requirement
F_BW	Ballast Weight Requirement
F_VE	Endurance Speed Requirement
F_TL	Total Length Requirement
F_ED	Endurance Duration Requirement
F_PR	Port Availability Requirement
F_VS	Sprint Speed Requirement

3.3 Objective Modules

Effectiveness, risk and cost are the three primary MOPs (or fitness functions) of this synthesis model. These attributes are different enough from one another that they cannot be rationally combined into a single overall measure. Since there are three objectives being compared, the non-dominated frontier will be a surface, as described in Section 2.2.

Risk and Effectiveness are two relatively abstract objectives that are based largely on expert opinion. The effectiveness of a few concepts can be analyzed using complex models or war-gaming exercises, but for an AUV model that has to rigorously analyze hundreds of models this is impractical. This synthesis model uses a methodology for calculating Overall Measure of Effectiveness (OMOE) and Overall Measure of Risk (OMOR) indices using expert opinion to synthesize diverse inputs such as mission requirements and experience.

3.3.1 Cost

This module calculates the total component purchase cost. Materials cost is included in this calculation. Man-hours and development time are not included. The Basic Cost of Construction (BCC) is the output of this module. It is calculated by taking the sum of the cost of components plus the materials cost:

$$BCC = 6W_{cap}(2.952) + AV_{mid}^{1+B} + \sum C_i \qquad (12)$$

where W_{cap} is the end cap weight, A and B are the constants which were found to fit the material type with its cost per volume. C_i is the component cost.

3.3.2 Overall Measure of Risk

The risk module calculates the Overall Measure of Risk (OMOR). The calculation considers three types of technology risk: performance, cost and schedule. Summing these three values of risk for applicable risk events and multiplying each type of risk by its associated weight factor results in the OMOR. Weight factors were determined using an analytical hierarchy process.

The Overall Measure of Risk (OMOR) function's purpose is to provide a quantitative measure of technology risk for a specific design based on the selection of components. These components are specified by DVs in Table 2. The calculation of the value of risk for any given variable, *I*, is the probability of failure, P_i, multiplied by the consequence of said failure, C_i, given in Equation 5.

$$R_i = P_i C_i \tag{13}$$

Risk events are associated with specific design variables. P_i and C_i are estimated using Table and Table. In order to be considered in the risk factors the event must have major impact on performance, cost or schedule.

Table 4: Event probability estimate

Probability	What is the Likelihood the Risk Event Will Occur?
0.1	Remote
0.3	Unlikely
0.5	Likely
0.7	Highly Likely
0.9	Near Certain

Table 5: Event consequence estimate

Consequence Level	Given the risk is realized, what is the Magnitude of impact? Performance	Schedule	
0.1	Minimal or no impact	Minimal or no impact	Minimal
0.3	Acceptable with some reduction in margin	Additional resources required; able to meet dates	<5%
0.5	Acceptable with significant reduction in margin	Minor slip in key milestones; not able to meet need date	5-7%
0.7	Acceptable; no remaining margin	Major slip in key milestone or critical path impacted	7-10%
0.9	Unacceptable	Can't achieve key team or major program milestone	>10%

Each event is then documented with its given value of risk and associated design variable or variables. Values for weight (W_{pe}, W_{sc} and W_{co}) were given to the three types of risk according to a pair-wise comparison by expert opinion on the subject (see Analytical Hierarchy Process, Section 2.3). The value for the OMOR was determined by using W_i, P_i, and C_i according to Equation 6.

$$OMOR = W_{pe}\frac{\sum P_i C_i}{\sum_i (P_i C_i)_{\max}} + W_{sc}\frac{\sum P_j C_j}{\sum_j (P_j C_j)_{\max}} + W_{co}\frac{\sum P_k C_k}{\sum_k (P_k C_k)_{\max}} \tag{14}$$

The weight factors and risk register is provided in Section 4.1.

3.3.3 Overall Measure of Effectiveness

This module calculates the measure of effectiveness for a specific design based on its Measures of Effectiveness (MOEs). The OMOE function must include all important effectiveness/performance attributes, both discrete and continuous, and ultimately be used to assess an unlimited number of AUV alternatives. Expert opinion is used to integrate these diverse inputs and assess the value or utility of the AUV MOEs. The approach used for this synthesis model is the Analytical Hierarchy Process (AHP), which was described in section 2.3.

The AHP in this situation is used to build an OMOE function. MOEs critical to AUV missions are identified with goal and threshold values for each. The MOEs are organized into an OMOE hierarchy and weights are found for each using pair-wise comparison and AHP. The weights for each MOE are used in the OMOE function as presented in Equation 15.

$$OMOE = \sum_{i=1}^{9}\left[\frac{\sum MOE_j}{\sum\left(MOE_j\right)_{\max}} W_i\right] + \sum_{i=10}^{13} W_i MOE_i \qquad (15)$$

where the MOE_j are pre-determined component Measures of Effectiveness for the 9 areas identified by the UUV Master Plan (US Navy, 2004). The MOE_i where i = 10 to 13 are calculation-based MOEs related to speed, duration and expandability. These are evaluated based on goals and thresholds determined prior to the synthesis model is run. If the calculated attribute is below the threshold, it is given a MOE value of 0, or if it is equal to or greater than the goal it is given a MOE of 1.0. Values in between the threshold and goal are scaled between 0 and 1. Goals are considered the point of diminishing return. Designs with attribute calculations greater than the goals don't provide an added benefit because the MOE will not go above 1.0. These goal and threshold values are Design Parameters. The Measure of Effectiveness tables for components are in Section 4.1.2.

Chapter 4 Optimization Setup and Results

4.1 Model Setup and Initialization

4.1.1 Components

There are three tab-delimited input files associated with the components. Components are the working parts of the AUV, whether they are electrical or mechanical in nature. The three files are:

Config.txt
Input.txt
OMOE.txt

The 'Input.txt' file is a master database of all of the components. It stores all of the component characteristics, which include:

- Component Name
- Component Type
- Payload
- Propulsion
- Electronic
- Power
- Communications
- Component Number
- Voltage Required
- Negative value if it receives
- Positive value if it provides
- Current Used/Provided
- Port Availability
- Negative value if it needs
- Positive value if it provides
- Ports:
- USB
- Firewire
- RS-232
- RS-485
- SPI
- I2C
- 1-wire
- Bluetooth
- 802.11 a/b/g/n (Wifi)
- Ethernet
- Fiberoptic
- Component Mass (in kg)
- Component X, Y, and Z, dimensions

- Component Center of Gravity Location (X, Y, and Z directions)
- Location in the AUV
- Nose
- Midsection
- Tail
- External
- Propulsion (Motor Rating)
- Added Drag (Input as C_{VAP} * S_{AP}, See Section 3.2.4)
- Component Cost

Table 6, below, is a simplified version of the 'Input.txt' file. The port columns have been merged to fit the table on the screen.

Table 6: Input.txt used in synthesis model (port columns merged to save space)

Component Name	Type	#	Voltage	Current	Ports	Mass	X	Y	Z	X CG	Y CG	Z CG	Loc	Prop	Drag	Cost
Depth sensor	1	1	-5	0.1	0	5.00	30	20	16	15.00	10.00	8.00	1	0	0	0
Power Board	3	2	-24	0.6	-1	191.50	111.86	69.9	16	55.93	34.95	8.00	2	0	0	0
CPU w/wifi w/CF	3	3	-5	1.3	2	126.50	95.77	90.19	20.25	47.89	45.10	10.13	2	0	0	200
Serial expansion board	3	4	0	0	8	70.30	106.8	95.92	11.33	53.40	47.96	5.67	2	0	0	0
Analog input	3	5	-5	0.02	-1	7.51	44.4	35.01	6.76	22.20	17.51	3.38	2	0	0	0
Acoustic Modem	5	6	-24	3	-1	286.70	215	44.53	33.2	80.00	34.53	16.60	2	0	0	0
Battery 1	4	7	12	8	0	516.90	200	50	30	100.00	25.00	15.00	2	0	0	200
Battery 2	4	8	24	4	0	513.00	190	50	30	95.00	25.00	15.00	2	0	0	200
RF Modem	5	9	-12	0.5	0	65.20	70	62	15	35.00	31.00	7.50	2	0	0	0
AHRS-IMU	1	10	-12	0.065	0	24.95	53	40	13	26.50	20.00	6.50	2	0	0	0
Antenna	5	11	0	0	0	3.97	105	12	12	52.50	6.00	6.00	4	0	0	0
PWM/Servo Controller	2	12	0	0	-1	3.30	23	23	10	11.50	11.50	5.00	3	0	0	0
Motor Controller	2	13	-12	0.5	0	22.68	45	25	8	22.50	12.50	4.00	3	0	0	0
RPM Sensor	1	14	-5	0.12	-1	10.63	38.01	25.48	6.59	19.01	12.74	3.30	3	0	0	0
Servo	2	15	0	0	0	3.97	30	23	10	15.00	11.50	5.00	3	0	0	0
Gear Box	2	16	0	0	0	38.27	50	30	5	25.00	15.00	2.50	3	0	0	0
Motor	2	17	0	0	0	46.49	28	28	25	14.00	14.00	12.50	3	240	0	60
Transducer	1	18	0	0	0	164.43	45	45	45	22.50	22.50	22.50	4	0	0	0
GPS	1	19	-5	0.06	-1	15.17	30.16	30.16	10.1	15.08	15.08	5.05	4	0	0	0
Moving Mass Servo	2	20	0	0	0	49.10	60	51	41	30.00	25.50	20.50	2	0	0	0
Alt Motor 1	2	21	-8	0	0	27.00	28	28	22	14.00	14.00	11.00	3	100	0	25
Alt Motor 2	2	22	-8	0	0	41.00	28	28	26	14.00	14.00	13.00	3	190	0	38
Alt Motor 3	2	23	-12	0	0	71.00	35	35	30	17.50	17.50	15.00	3	296	0	55
Alt Motor 4	2	24	-12	0	0	170.00	35	35	48	17.50	17.50	24.00	3	666	0	70
Alt Motor 5	2	25	-12	0	0	125.00	42	42	40	21.00	21.00	20.00	3	592	0	70
Alt Motor 6	2	26	-12	0	0	198.00	42	42	50	21.00	21.00	25.00	3	925	0	73
Alt Motor 7	2	27	-12	0	0	268.00	42	42	60	21.00	21.00	30.00	3	1110	0	75
Alt Motor 8	2	28	-32	0	0	635.00	63	63	62	31.50	31.50	31.00	3	1665	0	140
Alt Battery 1a	4	29	12	2.2	0	170.00	107	33	25	53.50	16.50	12.50	2	0	0	80
Alt Battery 1b	4	30	24	2.2	0	340.00	107	51	33	53.50	25.50	12.50	2	0	0	150
Alt Battery 2	4	31	12	5	0	388.00	160	48	25	80.00	24.00	12.50	2	0	0	180
Alt Battery 2b	4	32	24	5	0	726.00	160	50	45	80.00	25.00	22.50	2	0	0	300
Alt Battery 3	4	33	12	8	0	474.00	185	50	29	92.50	25.00	24.50	2	0	0	255
Alt Battery 4	4	34	12	1.9	0	108.00	107	32	16	53.50	16.00	8.00	2	0	0	60
ALT CPU 1	3	35	-5	1.3	2	150.00	96	90	30	48.00	45.00	15.00	2	0	0	563
ALT CPU 2	3	36	-5	0.116	4	83.00	96	90	22	48.00	45.00	11.00	2	0	0	174
Alt Payload 1	1	37	-5	1	0	50.00	50	30	20	25.00	15.00	10.00	2	0	0	75
Alt Payload 2	1	38	-5	0.75	0	50.00	50	20	20	25.00	10.00	10.00	2	0	0	75
Alt Payload 3	1	39	-5	0.5	0	50.00	50	20	20	25.00	10.00	10.00	2	0	0	75
Alt Payload 4	1	40	-5	1	0	50.00	50	20	20	25.00	10.00	10.00	2	0	0	75
Alt Payload 5	1	41	-5	1	0	50.00	40	30	20	20.00	15.00	10.00	2	0	0	50
Alt Payload 6	1	42	-5	0.5	0	50.00	40	20	20	20.00	10.00	10.00	2	0	0	50
Alt Payload 7	1	43	-5	0.5	0	50.00	30	30	20	15.00	15.00	10.00	2	0	0	50
Alt Payload 8	1	44	-5	0.5	0	50.00	30	20	20	15.00	10.00	10.00	2	0	0	50
Alt Payload 9	1	45	-5	0.5	0	50.00	60	20	20	30.00	10.00	10.00	2	0	0	100

The 'Config.txt' file organizes the components into configuration lists. It is separated by Component Type (Payload, Propulsion, Electronics, Power, Communications) and assigns Component Numbers to Configuration Numbers. For example, Payload Configuration 1 might use components 1, 3 and 5, while

Payload Configuration 2 uses components 2 and 4. Table 7, below, is the 'Config.txt' file used in this synthesis.

Table 7: Config.txt used in synthesis model (split into 4 sections to conserve space)

Type	Config #	Comp #
1	1	1
1	1	10
1	1	14
1	1	18
1	1	19
1	2	1
1	2	10
1	2	14
1	2	18
1	2	19
1	2	37
1	3	1
1	3	10
1	3	14
1	3	18
1	3	19
1	3	38
1	4	1
1	4	10
1	4	14
1	4	18
1	4	19
1	4	39
1	5	1
1	5	10
1	5	14
1	5	18
1	5	19
1	5	40
1	6	1
1	6	10
1	6	14
1	6	18
1	6	19
1	6	41
1	7	1
1	7	10
1	7	14
1	7	18
1	7	19
1	7	42
1	8	1
1	8	10
1	8	14
1	8	18
1	8	19
1	8	43

Type	Config #	Comp #
1	9	1
1	9	10
1	9	14
1	9	18
1	9	19
1	9	44
1	10	1
1	10	10
1	10	14
1	10	18
1	10	19
1	10	45
1	11	1
1	11	10
1	11	14
1	11	18
1	11	19
1	11	44
1	11	44
1	12	1
1	12	10
1	12	14
1	12	18
1	12	19
1	12	44
1	12	43
2	1	12
2	1	13
2	1	15
2	1	15
2	1	15
2	1	15
2	1	16
2	1	17
2	1	20
2	2	12
2	2	13
2	2	15
2	2	15
2	2	15
2	2	15
2	2	16
2	2	21
2	2	20

Type	Config #	Comp #
2	3	12
2	3	13
2	3	15
2	3	15
2	3	15
2	3	15
2	3	16
2	3	22
2	3	20
2	4	12
2	4	13
2	4	15
2	4	15
2	4	15
2	4	15
2	4	16
2	4	23
2	4	20
2	5	12
2	5	13
2	5	15
2	5	15
2	5	15
2	5	15
2	5	16
2	5	24
2	5	20
2	6	12
2	6	13
2	6	15
2	6	15
2	6	15
2	6	15
2	6	16
2	6	25
2	6	20
2	7	12
2	7	13
2	7	15
2	7	15
2	7	15
2	7	15
2	7	16
2	7	26
2	7	20

Type	Config #	Comp #
2	8	12
2	8	13
2	8	15
2	8	15
2	8	15
2	8	15
2	8	16
2	8	27
2	8	20
2	9	12
2	9	13
2	9	15
2	9	15
2	9	15
2	9	15
2	9	15
2	9	16
2	9	28
2	9	20
3	1	2
3	1	3
3	1	4
3	1	5
3	2	2
3	2	35
3	2	4
3	2	5
3	3	2
3	3	36
3	3	4
3	3	5
4	1	7
4	1	8
4	2	29
4	2	30
4	3	31
4	3	32
4	4	33
4	4	33
4	5	34
4	5	34
4	6	33
4	6	34
4	7	33
4	7	30
4	8	33
4	8	32
5	1	6
5	1	9
5	1	11

The last file is the 'OMOE.txt' file. This file includes the Effectiveness and Risk Registers for the component configurations. For each component type configuration (Ex. Payload Configuration 1, Power Configuration 3, etc) the 'OMOE.txt' file stores the 9 Effectiveness ratings from the UUV Master Plan (US Navy, 2004) and 3 risk types. The risk and effectiveness values were judged by expert opinion. Table 8, below is the 'OMOE.txt' file used in this synthesis.

Table 8: OMOE.txt used in AUV Synthesis Model

Config Type	Config #	ER1	ER2	ER3	ER4	ER5	ER6	ER7	ER8	ER9	Rperf	Rsched	Rcost
1	1	0.5	0.1	0.2	0.3	0.6	0.3	0.2	0.3	0.1	0.27	0.09	0.09
1	2	0.9	0.3	0.3	0.2	0.2	0.3	0.5	0.1	0.1	0.3	0.2	0.1
1	3	0.4	0.9	0.2	0.3	0.6	0.4	0.4	0.3	0.1	0.27	0.4	0.09
1	4	0.3	0.4	0.9	0.4	0.5	0.3	0.3	0.4	0.2	0.4	0.3	0.2
1	5	0.3	0.3	0.3	0.9	0.2	0.3	0.4	0.4	0.3	0.5	0.2	0.09
1	6	0.3	0.2	0.4	0.5	0.9	0.3	0.2	0.5	0.1	0.27	0.09	0.1
1	7	0.2	0.4	0.5	0.2	0.2	0.9	0.3	0.3	0.4	0.4	0.3	0.09
1	8	0.4	0.5	0.2	0.2	0.5	0.3	0.9	0.6	0.2	0.27	0.35	0.1
1	9	0.5	0.2	0.4	0.2	0.2	0.3	0.6	0.9	0.3	0.3	0.09	0.09
1	10	0.1	0.1	0.5	0.2	0.6	0.3	0.4	0.4	0.9	0.27	0.2	0.1
1	11	0.2	0.5	0.2	0.4	0.1	0.3	0.5	0.4	0.5	0.4	0.2	0.09
1	12	0.4	0.4	0.4	0.2	0.2	0.2	0.2	0.4	0.2	0.5	0.4	0.1
2	1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.45	0.25	0.15
2	2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.4	0.3	0.2
2	3	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.45	0.25	0.15
2	4	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.6	0.3	0.2
2	5	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.45	0.25	0.15
2	6	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.25	0.2
2	7	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.4	0.15
2	8	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.45	0.3	0.2
2	9	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.55	0.6	0.5	0.15
3	1	0.4	0.2	0.3	0.6	0.7	0.6	0.5	0.5	0.3	0.27	0.09	0.21
3	2	0.5	0.3	0.4	0.7	0.7	0.7	0.6	0.6	0.4	0.2	0.2	0.3
3	3	0.3	0.2	0.2	0.4	0.5	0.5	0.4	0.4	0.2	0.4	0.1	0.3
4	1	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.27	0.15	0.3
4	2	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.21
4	3	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.33	0.25	0.21
4	4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.2	0.3
4	5	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.3	0.15	0.21
4	6	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.4	0.2	0.3
4	7	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.35	0.2	0.25
4	8	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.6	0.5	0.21
5	1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.21	0.15	0.27

4.1.2 Design Parameter Initialization

Design Parameters are generated by the decision makers and define the characteristics about how the AUV operates and the environment it is contained within. The Risk and Effectiveness Weights are design parameters and have a large effect on how and if the AUV will survive to the next generation of the Genetic Algorithm. Table 9 includes the DPs used for this analysis. They were chosen based on expert experience and opinion.

Table 9: Design Parameter Initialization

DP Name	Value	Description
PC	0.6	Propulsive Coefficient
Eta	0.9	Motor Efficiency
Vsmin	5	Minimum Sprint Speed (knots)
Vsgoal	6	Goal Sprint Speed (knots)
Vemin	2.5	Minimum Endurance Speed (knots)
Vegoal	4	Goal Endurance Speed (knots)

MinDuration	4	Minimum AUV Duration at Endurance Speed (hours)
GoalDuration	6	Goal Endurance Duration (hours)
MinBallast	0	Minimum Ballast Mass (kg)
GoalBallast	0	Goal Ballast Mass (kg)
PR	0.47956	Performance Risk Weight
SR	0.11496	Schedule Risk Weight
CR	0.40548	Cost Risk Weight
EW1	0.19928	Effectiveness Weight (ISR)
EW2	0.17499	Effectiveness Weight (Oceanography)
EW3	0.10284	Effectiveness Weight (CN3)
EW4	0.028626	Effectiveness Weight (Mine Countermeasures)
EW5	0.023392	Effectiveness Weight (Anti-Submarine Warfare)
EW6	0.019135	Effectiveness Weight (Inspection/Identification)
EW7	0.019135	Effectiveness Weight (Payload Delivery)
EW8	0.066494	Effectiveness Weight (Information Operations)
EW9	0.017981	Effectiveness Weight (Time Critical Strike)
EW10	0.083953	Effectiveness Weight (Sprint Speed)
EW11	0.099519	Effectiveness Weight (Endurance Speed)
EW12	0.15633	Effectiveness Weight (Endurance Duration)
EW13	0.0083279	Effectiveness Weight (Ballast/Expandability)

Based on a survey evaluated by experts, the risk importance and effectiveness importance are broken down according to Figure 7 and Figure 8 respectively.

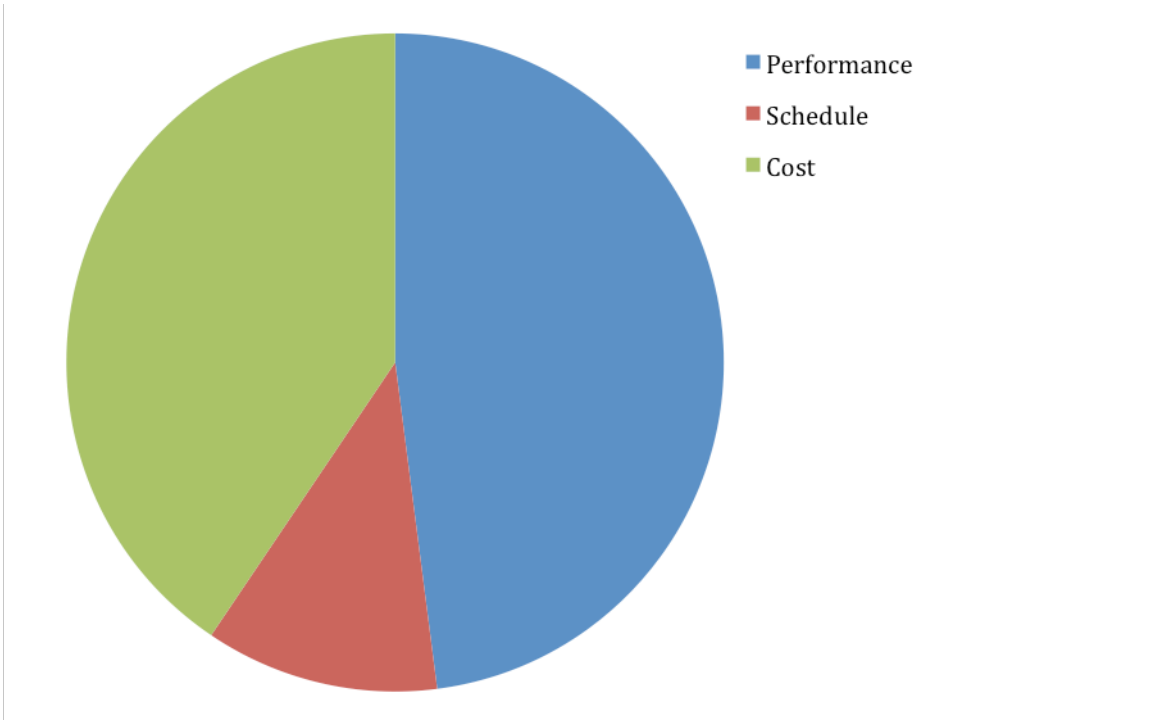


Figure 7: Risk Importance Breakdown

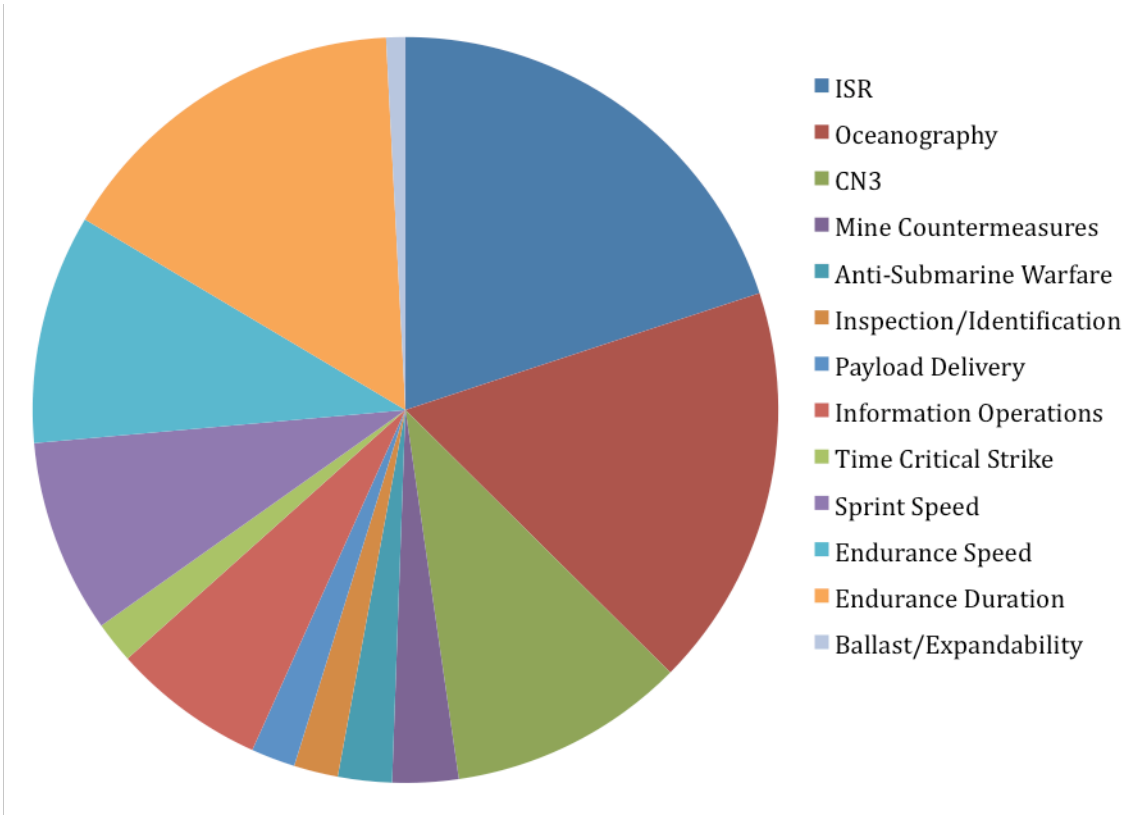


Figure 8: Effectiveness Importance Breakdown

4.1.3 Design Variable Setup

Design Variables make up the DNA of the AUV. The MOGO varies each DV between its lower and upper bounds. Table 10 lists the lower and upper bounds considered for this synthesis.

Table 10: Design Variable Initialization

DV Name	Lower	Upper	Description
D	0.05	1.0	Vehicle Diameter (m)
LtoD	1.0	20.0	Length to Diameter Ratio
BtoD	1.0	1.0	Beam to Diameter Ratio
nf	0.3	0.9	Forward shape coefficient
na	0.3	0.9	Aft shape coefficient
Ve	2.0	5.0	Endurance Speed
CommConf	1	1	Communication Configuration
PayConf	1	12	Payload Configuration
PropConf	1	9	Propulsion Configuration
BatConf	1	8	Battery Configuration
ElecConf	1	3	Electronics Configuration
WallType	1	4	Hull Wall Thickness/Material

4.1.4 Constraints and Objectives

As discussed in Section 3.2.5, all feasibility constraints must be greater than or equal to zero for the design to be feasible. *F_TL,* the total length constraint, and *F_TB,* the total ballast constraint, also have upper constraints. *F_TL* requires the total compressed length of the AUV to be less than 80% of the length of the AUV overall. *F_TB* requires that the density of the total ballast must be below the density of brass, or 8750 kg/m^3, this ensures that a reasonable material will be used to ballast the AUV.

Once the constraints are setup, the objectives are set up. For this analysis the variables for Cost, BCC, and Risk, OMOR, are to be minimized by the MOGO. The Effectiveness variable, OMOE, is to be maximized.

4.1.5 Genetic Algorithm Setup

Once the components, DPs, DVs, constraints, and objectives are defined, the genetic algorithm parameters are entered. Figure 9 is the Optimization Parameter window used by the MOGO module. For this analysis a population size of 200 was chosen that would preserve 50 designs to be parents for the next generation. The convergence method chosen was to allow the model to go 50 generations without improvement before stopping.

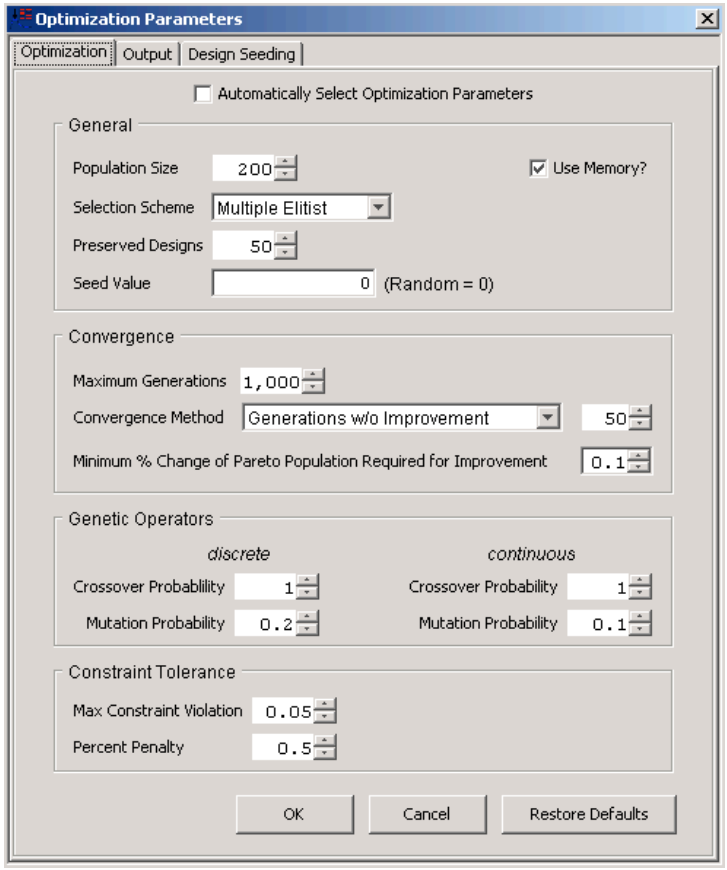


Figure 9: Genetic Algorithm Optimization Parameters

4.2 Optimization Results

Figure 10 is the three dimensional view of the MOP design space. Each point on these graphs represents an AUV design. Figure 4a shows the entire final population. The Pareto designs are shown in Figure 4b. Gray points are infeasible designs.

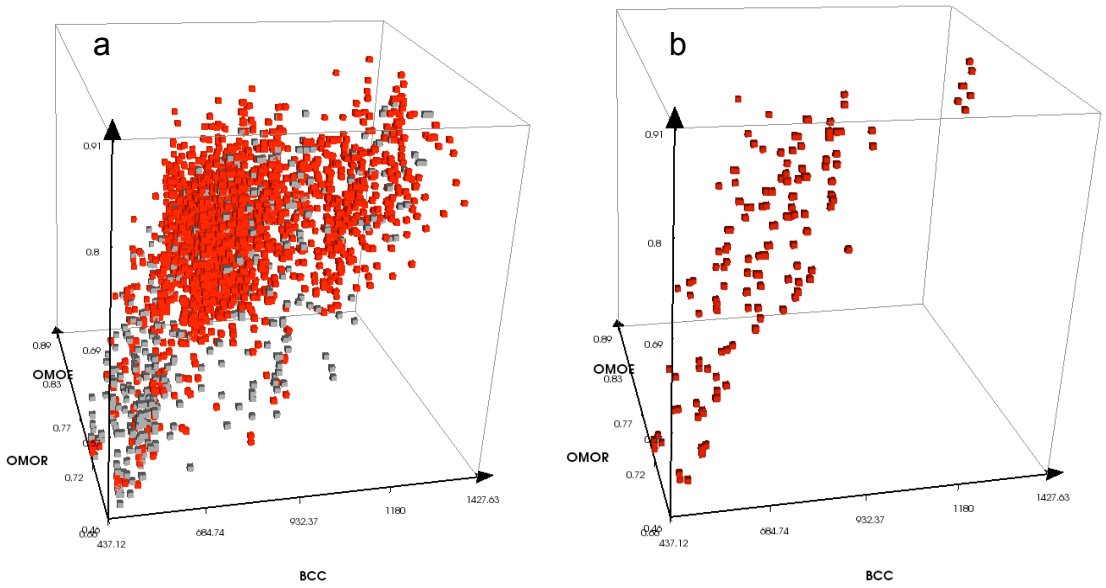


Figure 10: 3D Measure of Performance Space; a) entire final population, b) Pareto Designs

The Pareto front shows that, generally, as Cost and Risk go up, the Effectiveness also goes up. This trend is stronger for cost than for risk as can clearly be seen in the two-dimensional plots below. The relationship between cost and risk among the Pareto designs is less well defined.

The following three figures compare the three MOPs against one another. The graphs have one Measure of Performance on each of the two axes, and are colored based on the third unused MOP. The Pareto designs are highlighted using black plus signs. Figure 11 plots Effectiveness on the X-axis with Cost on the Y-axis. Figure 12 shows Risk on the X-axis with Cost on the Y-axis, and Figure 13 plots Effectiveness on the X-axis and Risk on the Y-axis.

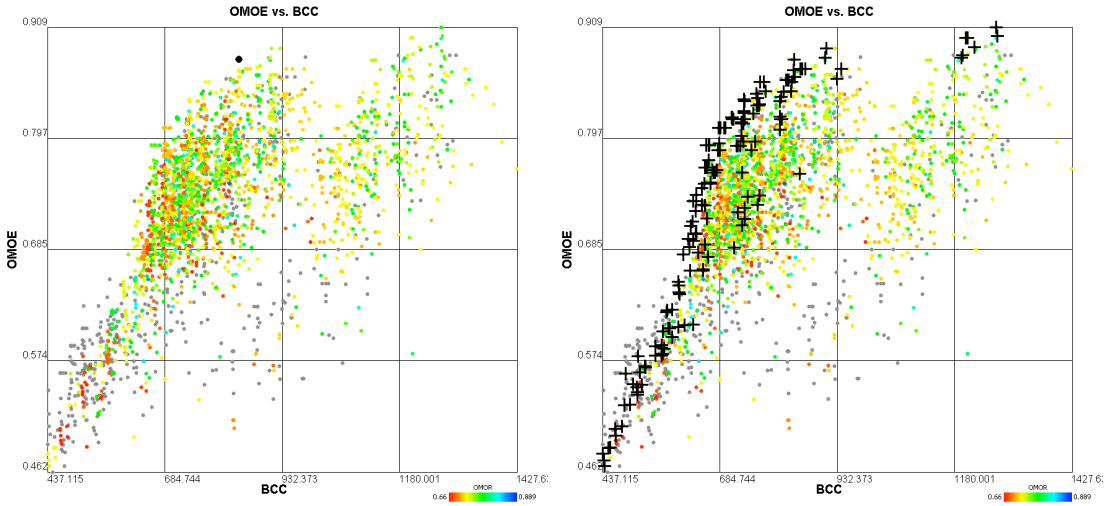


Figure 11: Cost vs. Effectiveness, Risk Colored; Pareto Designs Highlighted (right)

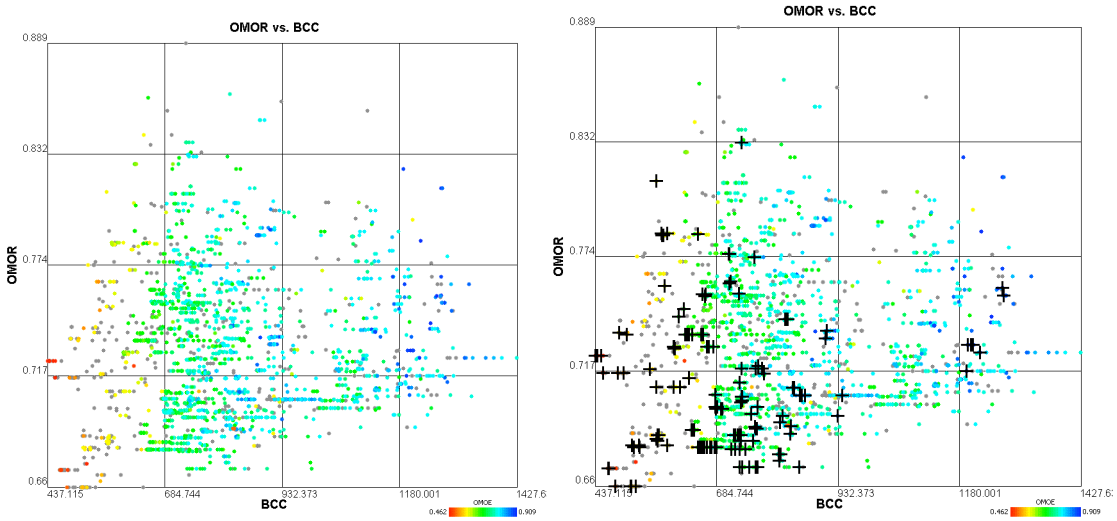


Figure 12: Cost vs. Risk, Effectiveness Colored; Pareto Designs Highlighted (right)

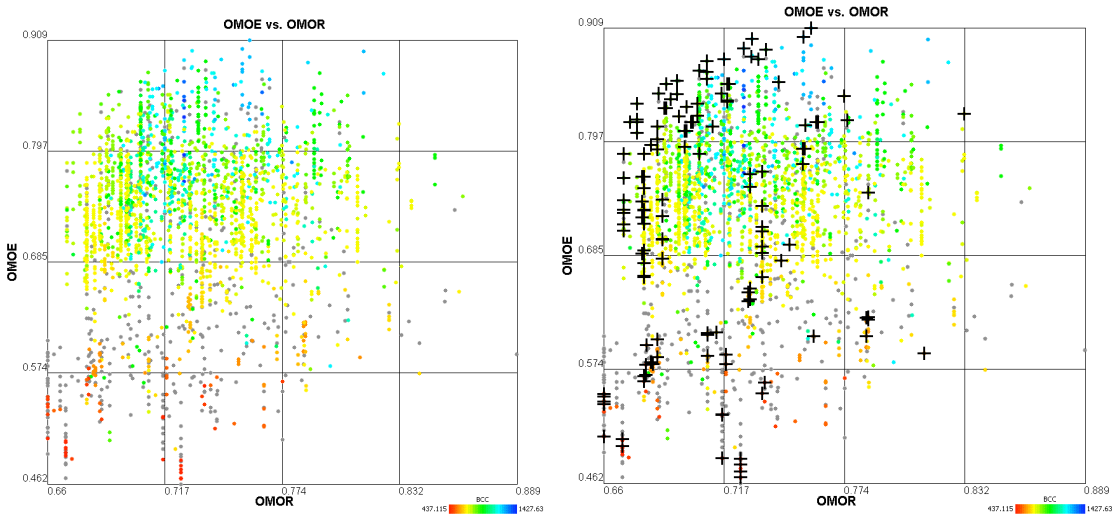


Figure 13: Risk vs. Effectiveness, Cost Colored; Pareto Designs Highlighted (right)

Looking at the Pareto front for Figure 11, as Effectiveness goes up Cost also goes up, almost linearly. There isn't a discernable pattern for the Cost vs Risk and Risk vs Effectiveness plots.

The next step for the decision maker(s) is to choose candidate designs. "Knees in the curve" are generally used to choose designs. "Knees" are the locations on the Pareto front where significant changes in the slope occur. For example, in Figure 14, looking at Effectiveness vs. Cost two trend lines have been added that indicate the relative slope of the Pareto front. The location where the two trend lines intersect would be a "knee" in the curve. With line "A" a little more cost will buy a lot more effectiveness, but for line "B" the cost of more effectiveness is much higher. Section 5.2 uses this method to find the five candidate AUVs generated from the synthesis model.

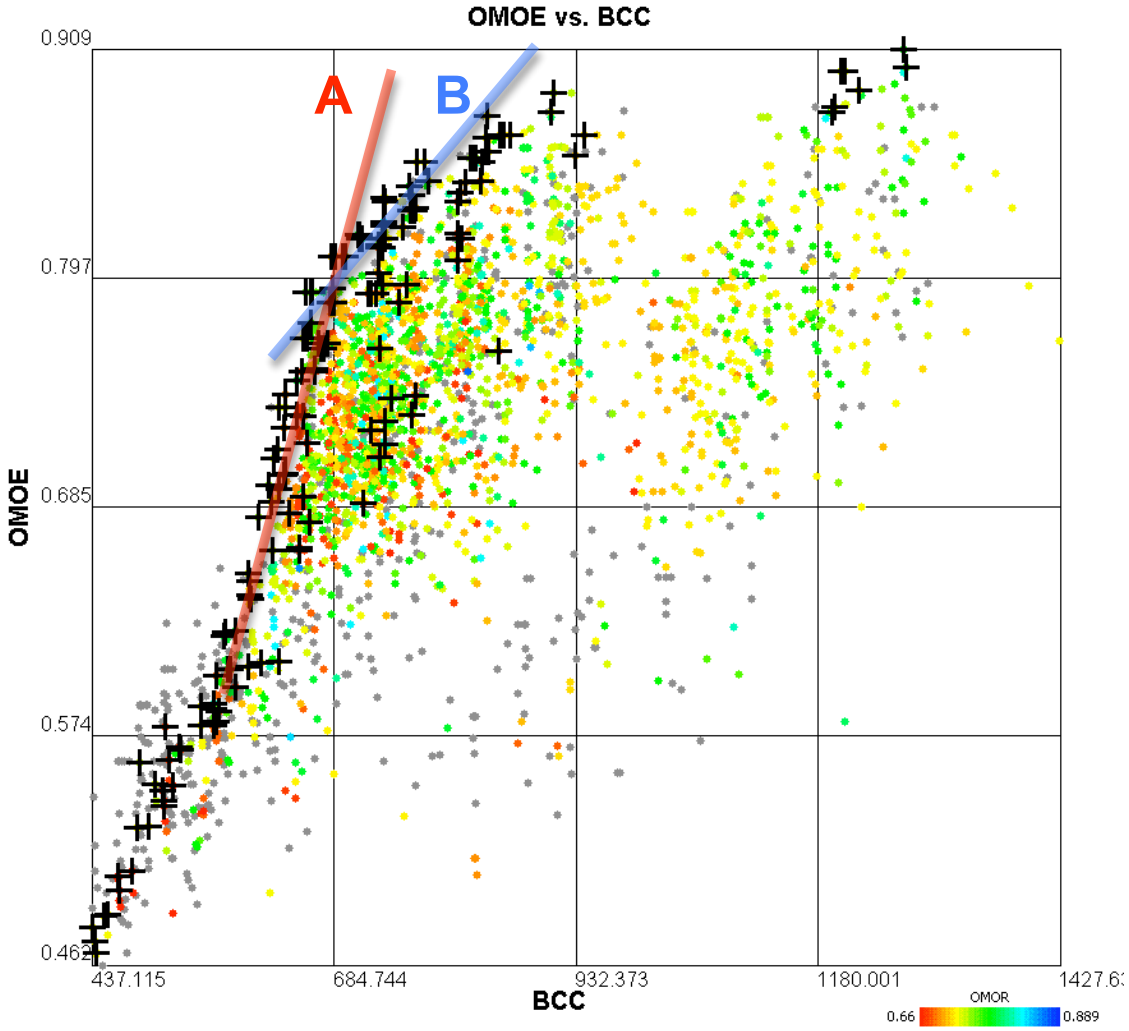


Figure 14: OMOE vs. BCC with trend lines indicating a "Knee in the curve"

Chapter 5 Design Variable and AUV Studies

5.1 Design Variable Study

ModelCenter has a built-in data explorer that includes a Variable Influence Profiler. A part of this profiler is the Main Effects plot, which can be drawn for each design parameter. It shows quantitatively how the selected output variable changes (on average) as the design parameter is varied from its lower bound to its upper bound. The influence of all the other design parameters is averaged out. The designs from the last generation available are used for this analysis. The following three figures show how the Design Variables affect the Measures of Performance of the AUV.

Figure 15 shows the Main Effects on OMOE. The Length to Diameter Ratio, *LtoD*, and Propulsion Configuration are the primary items that affect the Effectiveness of the AUV. These two variables both affect the Speed and Duration aspects of the Overall Measure of Effectiveness. *LtoD* has an affect on the resistance of the AUV, and the Propulsion Configuration selects the motor to be used, which can affect the Sprint Speed and Endurance Duration of the AUV.

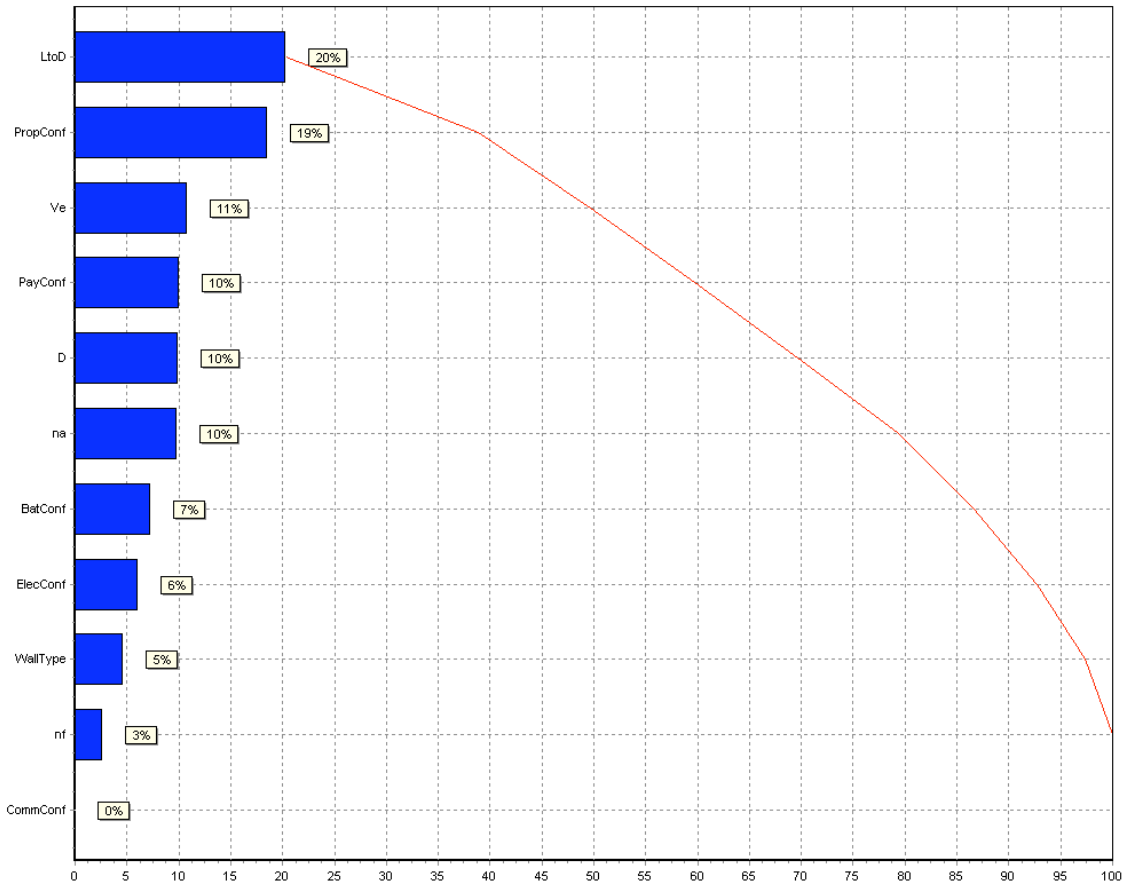


Figure 15: Main Effects on OMOE from last generation of model synthesis

Figure 16 shows the Main Effects on OMOR. The Electronics Configuration is the primary DV that affects the Risk of the AUV. The DVs having the next greatest effect on Risk are the AUV Diameter, *D*, Length to Diameter Ratio, *LtoD*, and the Propulsion Configuration. The Diameter and Length to Diameter Ratio can affect what Electronic Components could fit into the AUV. The component configurations are the only things associated with Risk, and the Electronic and Propulsion configurations have the highest potential risks associated with them. The other DVs may force or eliminate the selection of certain components because of size or weight constraints.

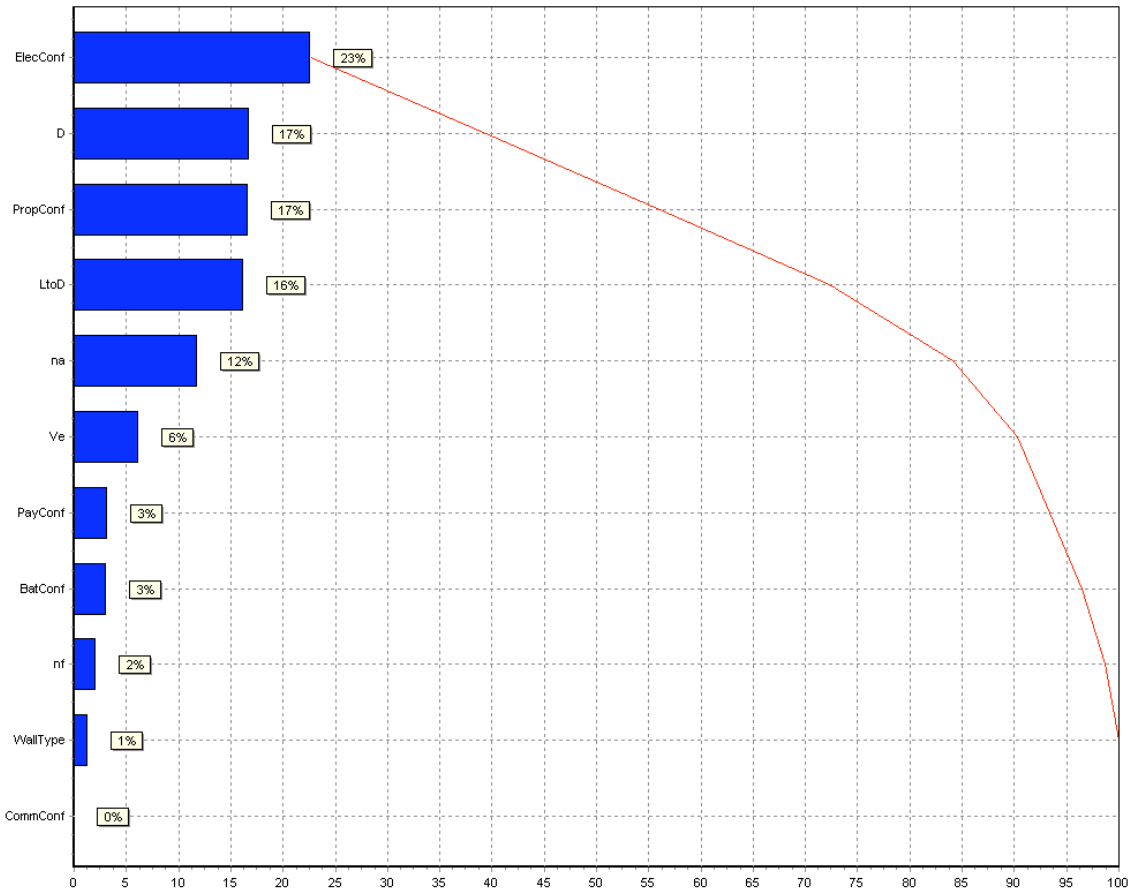


Figure 16: Main Effects on OMOR from last generation of model synthesis

Figure 17 shows the Main Effects on Cost. Diameter, *D*, Length to Diameter ratio, *LtoD*, and the Wall Type are the three major DVs that drive the cost, accounting for 62% of the Main Effect of Cost. This would imply that the hull material and cost of material are the principle contributors. Large diameters and length to diameter ratios would lead to the requirement of more hull material, which would lead to a greater cost.

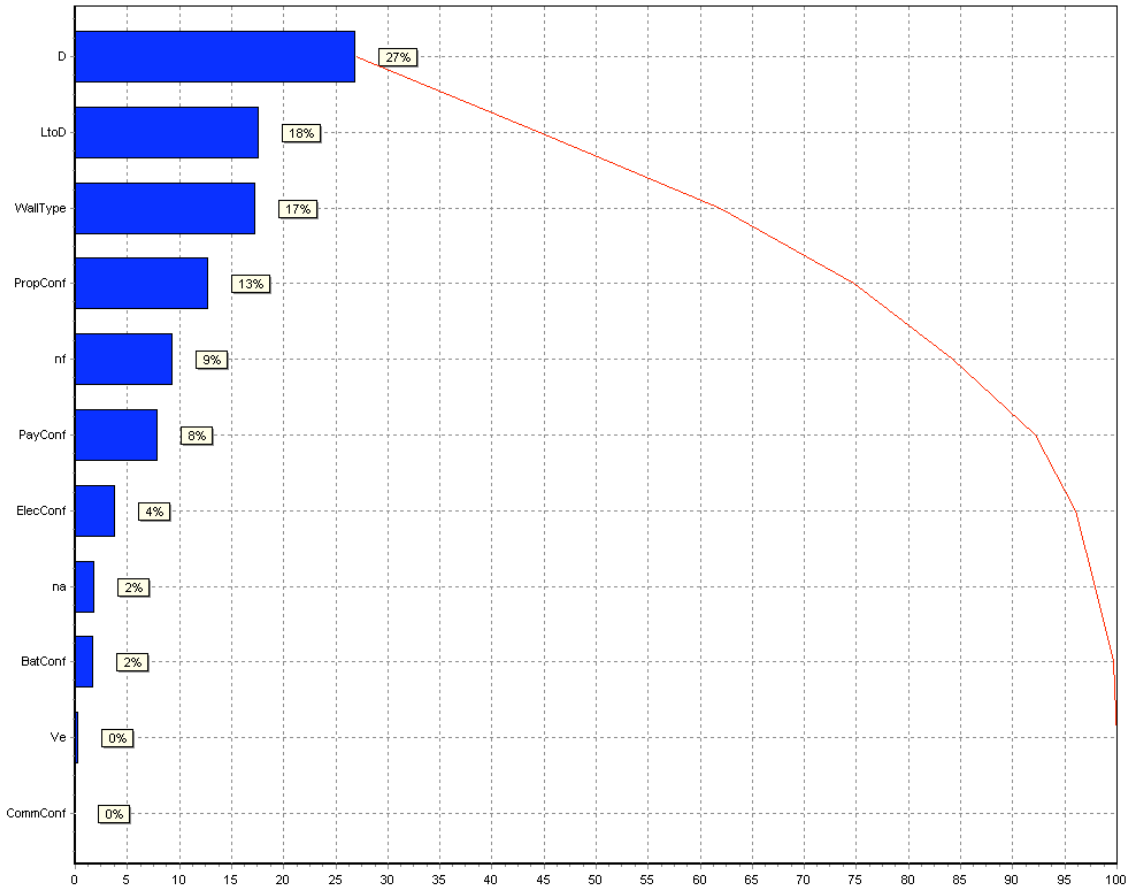


Figure 17: Main Effects on Cost from last generation of model synthesis

Comparing the diameter histogram in Figure 20 to the OMOE and OMOR diameter comparisons in Figure 18 and Figure 19, the Pareto designs seem to coincide with the most frequently occurring diameter range. Diameters approximately between 0.11 m and 0.13 m have a high concentration of Pareto-Frontier designs. This indicates that the genetic algorithm is converging to an optimum diameter and using it frequently in parent designs. The *LtoD*, n_f, n_a and V_e DVs have similar results, with there being a higher concentration of Pareto designs at the peaks of their respective histograms.

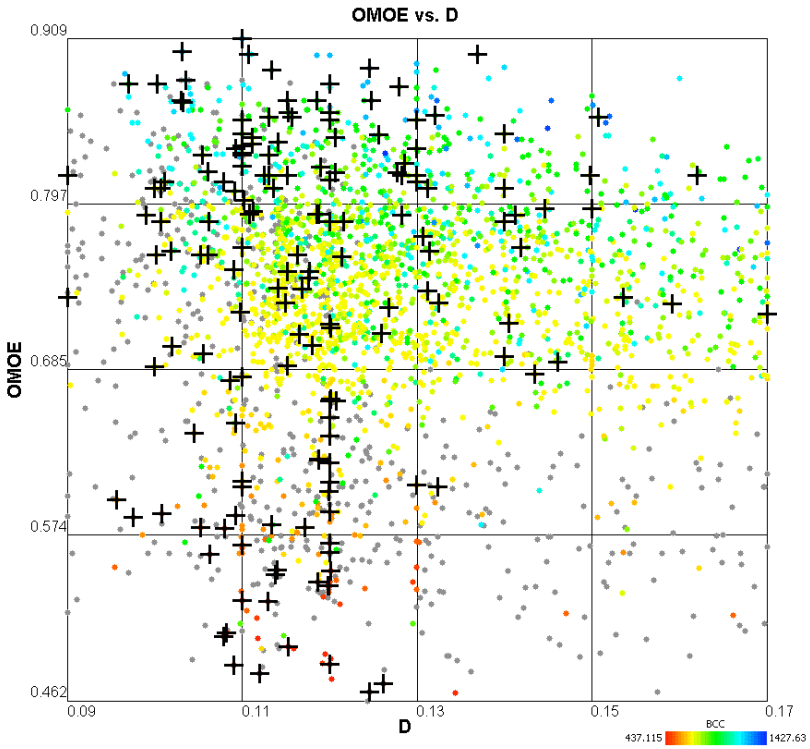


Figure 18: Diameter vs. OMOE, Cost Colored; Pareto Designs Highlighted

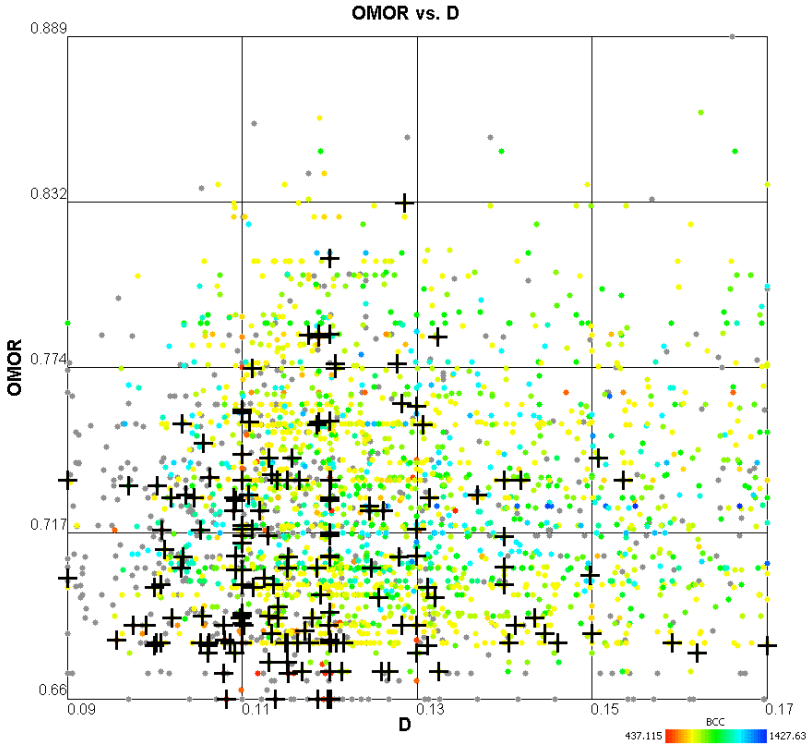


Figure 19: Diameter vs. OMOR, Cost Colored; Pareto Designs Highlighted

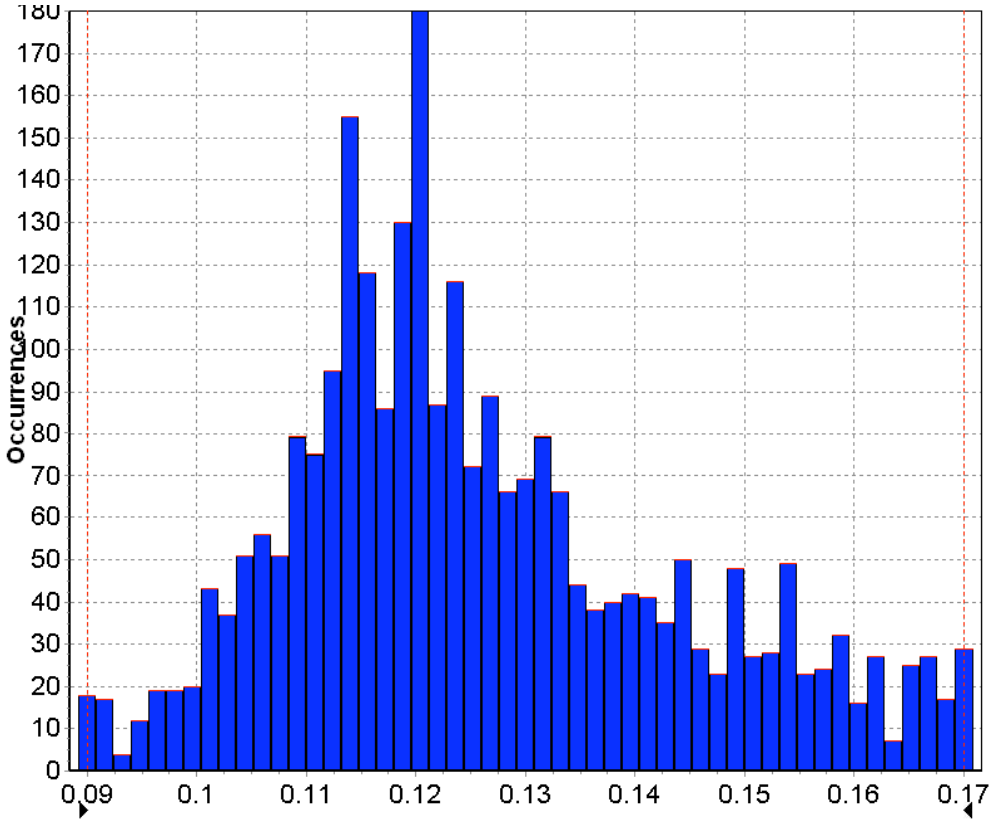


Figure 20: Diameter Histogram generated from the design space generated by the synthesis model

5.2 Comparison of Five AUVs from the Pareto Front

Five AUVs were chosen from the Pareto Front to compare and analyze. All designs on the Pareto Front are feasible. Below, in Figure 21, the five selected designs have been highlighted on the 3D Measure of Performance graph. Table 11 lists their DVs, and Table 12 lists their MOP and Constraint results. In Table 12, the extremes for each of the MOP and Constraint values among the chosen designs are highlighted green and red for best and worst respectively.

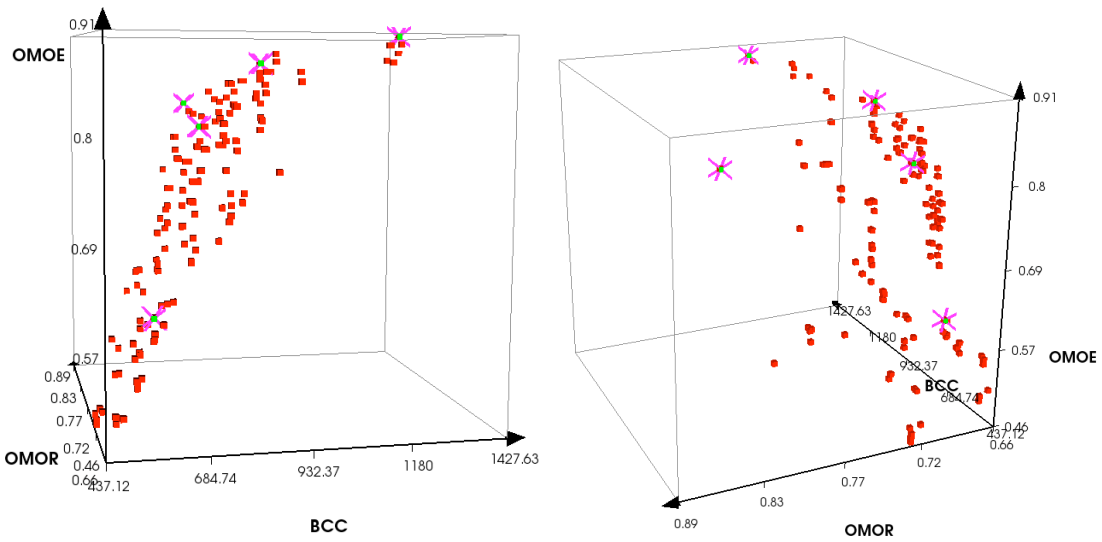


Figure 21: Selected Pareto AUVs generated from Synthesis Model, viewed from different angles

Table 11: Design Variables for Selected Pareto Front AUVs

Design Number:	2410	1539	1604	321	92
D (m)	0.128	0.114	0.129	0.120	0.110
LtoD	7.93	7.87	9.28	6.68	8.77
BtoD	1.0	1.0	1.0	1.0	1.0
nf	0.3877	0.3734	0.3443	0.3083	0.3
na	0.5977	0.4825	0.3698	0.3398	0.3
Ve (knots)	4.297	4.233	4.363	3.229	4.5
CommConf	1	1	1	1	1
PayConf	3	1	4	9	3
PropConf	8	7	7	8	9
BatConf	3	1	1	2	3
ElecConf	1	1	3	1	2
WallType	1	1	1	1	1

The Design Variables of the five selected AUVs in Table 11 are similar. The Diameters are all approximately 0.12 m. The Length to Diameter ratios have a larger range, going from 6.68 to 9.28. The forward shape coefficient, n_f, are reasonably diverse, but remain towards the lower end of the possible range. The aft shape coefficient, n_a, are more diverse than the forward shape coefficient, but also remain towards the lower end of the possible range.

The component configurations are diverse. No two designs had the same component configurations. Each selected design has a different payload configuration. Propulsion and battery configurations have two sets of repeating configurations. The propulsion configurations all use the most powerful motors available in the 'input.txt' component database, and the battery configurations

generally are the batteries with the highest capacity. The first electronic component configuration was used most often, but there are only three electronic configurations available to choose from.

The Wall Type of the AUVs are all the same. All of the AUVs selected the 0.0625 in thick polycarbonate material because it was the cheapest. This Synthesis Model does not account for structures which would have an affect on material selection (See Section 6.1 Future Work).

Table 12: Measure of Performance and Constraint Results for Selected Pareto Front AUVs

Design Number:	2410	1539	1604	321	92
OMOE	0.8761	0.8077	0.8244	0.6033	0.9090
OMOR	0.7087	0.6990	0.8313	0.6850	0.7584
BCC	841.64	682.43	735.15	563.87	1267.59
F_TB	704.17	683.00	802.37	814.80	745.25
F_BW	3.92	2.21	5.22	2.74	2.14
F_TL	0.6215	0.7380	0.5026	0.7482	0.7149
F_VE	0.7188	0.6932	0.7456	0.2916	0.8
F_VS	3.31	3.41	2.93	3.83	4.49
F_PR	0.667	0.667	1.5	0.667	1.667
F_ED	0.642	0.961	0.664	0.014	0.746

Designs 2410 and 1539 have MOPs that fall in the midrange of the five selected designs. The only major differences in their design variables are the Payload, Propulsion and Battery configurations. Design 1539 has less expandability (less Ballast Weight) but it also has a lower cost.

Design 1604 has the best expandability of all the designs, but it also has the highest risk and slowest sprint speed. The sprint speed is still above the threshold. The compressed length feasibility constraint is also the lowest, which might infer that this would be the easiest design to arrange. 5.22 kg of mass are required to ballast this vehicle, but this mass could take the form of additional payload.

Design 321 has the lowest risk and cost, but is the least effective design. It also has the slowest endurance speed of the five selected designs. Arrangement may also be a factor because the hull arrangement algorithm calculated this design to have the worst compressed length amongst the five chosen. It has the *F_TL* closest to 80% of the total overall length, which is the upper bound for the constraint.

Design 92 has the highest effectiveness of the five chosen designs, but it also the most expensive. It only has 2.14 kg of Ballast Weight available for additional payload, but also the most ports available for payload. This design has the highest endurance and sprint speeds.

With the potential designs chosen, it is now up to the decision-maker(s) to choose the design that best fits their preferences. The Measures of Performance need to be carefully weighed against the vehicle attributes, which include the DVs and the results of the Feasibility/Constraint analysis. The decision-maker(s) also need to take into account the limitations of the model with respect to the structural performance of the AUV and the controllability of the AUV. Both are beyond the scope of this project (See Section 6.2, Future Work).

Chapter 6 Conclusion

The goal of the concept design process is to identify feasible, non-dominated concepts so the decision-makers can base their selection on the objective attributes. There should be no bias for particular vehicle characteristics. Vehicle characteristics are only fundamental and intermediate parameters that lead to the calculation of the objective attributes of the AUV.

Multi-Disciplinary Optimization is essential for the design of highly integrated systems, such as AUVs, because it integrates multiple disciplines so that effective system-wide decisions can be made. The optimal system design is often something unique or non-intuitive and by using MDO it increases the confidence that the optimal design variables have been used to achieve the best design possible.

The MDO is greatly sped up using a genetic algorithm because it intelligently searches the design space instead of using a systematic or random brute force approach. Genetic algorithms are most appropriate to optimizing AUV designs because the AUVs are highly discontinuous, disjointed functions where no closed-form function exists.

Once the synthesis model has generated a solution of designs it is easy for the decision makers to choose potential candidates for further evaluation. The synthesis model does the heavy lifting of the analysis work cutting down millions of design combinations into tens of designs in the form of a Pareto front. Looking at the “knees” in the Pareto front allows the decision makers to narrow this down even further. With the final candidate designs more a more refined analysis can be performed using external tools or experience.

6.1 Future Work

The following could be more closely integrated into the synthesis model:
Beam to Diameter Ratio Effects
Computational Fluid Dynamics (CFD)
Structures
Dynamics and Control
A more refined Arrangement algorithm
Propeller design/optimization

The Beam to Diameter ratio was set to 1.0 for the example AUV Synthesis. Arrangement and Resistance modules would need to be modified to work with non-circular cross sections.

The arrangement algorithm used in this synthesis model is very basic. Optimally the algorithm should arrange the entire AUV and also optimize the placement of objects to optimize the center of gravity of the AUV (which also contributes to the controllability of the AUV).

Computational Fluid Dynamics could be used to provide better power and drag estimates, and possibly an analysis that could assist in vehicle dynamics and control analysis. The major drawback with computational fluid dynamics is that it is very computationally intensive and wouldn't allow for realistic development times. A way around this issue is to use response surface calculations instead. This would involve performing CFD analysis round a certain design space and fitting equations to match the data. Controls design/analysis is a major component of AUV design. Controllability of the AUV could tie into the Risk and/or Effectiveness of the AUV.

Structural analysis is another AUV design component that is lacking. The depth attainable by AUVs is unknown and may be very important to decision makers. Finite-Element Models could be used to add in the structural analysis, but this would also increase the computation time for each model. Response surface models could also be used for this, but it may be easier to develop a model based around structural weight and attainable depth. Depth attainability could be integrated into both the Risk and Effectiveness Measures of Performance.

Propeller design and optimization could be integrated directly into the resistance module of the synthesis model. Two approaches are the most likely to be implemented. Approach 1 would optimize the propeller for the AUV's sprint speed and endurance velocity. Approach 2 would have a database of propellers and the propeller would be a part of the configuration files. Approach 1 would be more computationally intensive than Approach 2.

References

Brown, A., Thomas, M., "Reengineering the Naval Ship Concept Design Process." ASNE From Research to Reality in Ship systems Engineering Symposium, Tysons Corner, VA, September 1998.

Brown, A., Salcedo, J. "Multiple Objective Genetic Optimization in Naval Ship Design", ASNE Day 2002, 2002.

Brown, A., Salcedo, J. "Multiple Objective Genetic Optimization in Naval Ship Design", *Naval Engineers Journal*, Vol. 115, No. 4, pp. 49-61, 2003.

Brown, A. "Submarine Design Notes." Virginia Tech Aerospace and Ocean Engineering Department, 2007.

Chen, Y., "Formulation of a Multi-Disciplinary Design Optimization of Containerships.", MS Thesis, Department of Aerospace and Ocean Engineering, Virginia Tech, 1999.

Demko, D., "Tools for Multi-Objective and Multi-Disciplinary Optimization in Naval Ship Design.", MS Thesis, Department of Aerospace and Ocean Engineering, Virginia Tech, 2005.

Gillmer, T.C., Johnson, B. *Introduction to Naval Architecture*, US Naval Institute Press; 2nd print with revisions, December 1982.

Good, N., Brown, A., "Multiple-Objective Concept Design of an Advanced Logistics Delivery System Ship (ALDV)." ASNE Joint Sea Basing Symposium, Arlington, VA, March 2006

Jackson, H. A., P.E., CAPT USN (Ret.). MIT Professional Summer Course "Submarine Design Trends", 1992.

Klasen, E. "Confidence of Success in Multi-Criteria Optimization of Multi-Disciplinary Ship Design Models." VPISU Report (2005)

Lewis, Edward V., editor, *Principles of Naval Architecture*, second revision, Society of Naval Architects and Marine Engineers, 1988.

Maines, A., Martz, M. Ovren, J., Palmer, M., Sloan, A., Story, W., "Design Report: Large Ocean Interface Submarine SSLOI." 2007 ASNE/SNAME Dr. James A. Lisnyk Ship Design Competition. 2007.

Mierzwicki, T., "Risk Index for Multi-objective Design Optimization of Naval Ships", MS Thesis, Department of Aerospace and Ocean Engineering, Virginia Tech, 2003.

Mierzwicki, T., Brown, A., "Risk Metric for Multi-Objective Design of Naval Ships", *Naval Engineers Journal*, Vol. 116, No. 2, pp. 55-71, 2004.

Model Center Software help files, Phoenix Integration, Blacksburg, VA, 2004.

"Monte Carlo method.", Wikipedia, <http://en.wikipedia.org/wiki/Monte_carlo_method>, May 10, 2008.

Navy, U.S., "The Navy Unmanned Undersea Vehicle (UUV) Master Plan", US Navy, November 9, 2004.

"Particle swarm optimization.", Wikipedia, <http://en.wikipedia.org/wiki/Particle_swarm_optimization>, May 10, 2008.

Saaty, T.L. *The Analytical Hierarchy Process*, RWS Publications, Pittsburg, 1996.

Stepanchick, J., Brown, A., “Revisiting DDGX/DDG-51 Concept Exploration.” ASNE Day, Arlington, VA, June 2006

Appendix

A. AHP Effectiveness Code

```
function AHPEtest()
%% Analytical Heirarchy Process Script

AHPEffectiveness = ones(13);
in = input('ISR vs Oceanography (1-9): ');
AHPEffectiveness = AHPE(1,2,in,AHPEffectiveness);
in = input('ISR vs CN3 (1-9): ');
AHPEffectiveness = AHPE(1,3,in,AHPEffectiveness);
in = input('ISR vs MCM (1-9): ');
AHPEffectiveness = AHPE(1,4,in,AHPEffectiveness);
in = input('ISR vs ASW (1-9): ');
AHPEffectiveness = AHPE(1,5,in,AHPEffectiveness);
in = input('ISR vs II (1-9): ');
AHPEffectiveness = AHPE(1,6,in,AHPEffectiveness);
in = input('ISR vs PD (1-9): ');
AHPEffectiveness = AHPE(1,7,in,AHPEffectiveness);
in = input('ISR vs IO (1-9): ');
AHPEffectiveness = AHPE(1,8,in,AHPEffectiveness);
in = input('ISR vs TCS (1-9): ');
AHPEffectiveness = AHPE(1,9,in,AHPEffectiveness);
in = input('ISR vs Vs (1-9): ');
AHPEffectiveness = AHPE(1,10,in,AHPEffectiveness);
in = input('ISR vs Ve (1-9): ');
AHPEffectiveness = AHPE(1,11,in,AHPEffectiveness);
in = input('ISR vs Ed (1-9): ');
AHPEffectiveness = AHPE(1,12,in,AHPEffectiveness);
in = input('ISR vs B (1-9): ');
AHPEffectiveness = AHPE(1,13,in,AHPEffectiveness);

in = input('Oceanography vs CN3 (1-9): ');
AHPEffectiveness = AHPE(2,3,in,AHPEffectiveness);
in = input('Oceanography vs MCM (1-9): ');
AHPEffectiveness = AHPE(2,4,in,AHPEffectiveness);
in = input('Oceanography vs ASW (1-9): ');
AHPEffectiveness = AHPE(2,5,in,AHPEffectiveness);
in = input('Oceanography vs II (1-9): ');
AHPEffectiveness = AHPE(2,6,in,AHPEffectiveness);
in = input('Oceanography vs PD (1-9): ');
AHPEffectiveness = AHPE(2,7,in,AHPEffectiveness);
in = input('Oceanography vs IO (1-9): ');
AHPEffectiveness = AHPE(2,8,in,AHPEffectiveness);
in = input('Oceanography vs TCS (1-9): ');
AHPEffectiveness = AHPE(2,9,in,AHPEffectiveness);
```

```
in = input('Oceanography vs Vs (1-9): ');
AHPEffectiveness = AHPE(2,10,in,AHPEffectiveness);
in = input('Oceanography vs Ve (1-9): ');
AHPEffectiveness = AHPE(2,11,in,AHPEffectiveness);
in = input('Oceanography vs Ed (1-9): ');
AHPEffectiveness = AHPE(2,12,in,AHPEffectiveness);
in = input('Oceanography vs B (1-9): ');
AHPEffectiveness = AHPE(2,13,in,AHPEffectiveness);

in = input('CN3 vs MCM (1-9): ');
AHPEffectiveness = AHPE(3,4,in,AHPEffectiveness);
in = input('CN3 vs ASW (1-9): ');
AHPEffectiveness = AHPE(3,5,in,AHPEffectiveness);
in = input('CN3 vs II (1-9): ');
AHPEffectiveness = AHPE(3,6,in,AHPEffectiveness);
in = input('CN3 vs PD (1-9): ');
AHPEffectiveness = AHPE(3,7,in,AHPEffectiveness);
in = input('CN3 vs IO (1-9): ');
AHPEffectiveness = AHPE(3,8,in,AHPEffectiveness);
in = input('CN3 vs TCS (1-9): ');
AHPEffectiveness = AHPE(3,9,in,AHPEffectiveness);
in = input('CN3 vs Vs (1-9): ');
AHPEffectiveness = AHPE(3,10,in,AHPEffectiveness);
in = input('CN3 vs Ve (1-9): ');
AHPEffectiveness = AHPE(3,11,in,AHPEffectiveness);
in = input('CN3 vs Ed (1-9): ');
AHPEffectiveness = AHPE(3,12,in,AHPEffectiveness);
in = input('CN3 vs B (1-9): ');
AHPEffectiveness = AHPE(3,13,in,AHPEffectiveness);

in = input('MCM vs ASW (1-9): ');
AHPEffectiveness = AHPE(4,5,in,AHPEffectiveness);
in = input('MCM vs II (1-9): ');
AHPEffectiveness = AHPE(4,6,in,AHPEffectiveness);
in = input('MCM vs PD (1-9): ');
AHPEffectiveness = AHPE(4,7,in,AHPEffectiveness);
in = input('MCM vs IO (1-9): ');
AHPEffectiveness = AHPE(4,8,in,AHPEffectiveness);
in = input('MCM vs TCS (1-9): ');
AHPEffectiveness = AHPE(4,9,in,AHPEffectiveness);
in = input('MCM vs Vs (1-9): ');
AHPEffectiveness = AHPE(4,10,in,AHPEffectiveness);
in = input('MCM vs Ve (1-9): ');
AHPEffectiveness = AHPE(4,11,in,AHPEffectiveness);
in = input('MCM vs Ed (1-9): ');
AHPEffectiveness = AHPE(4,12,in,AHPEffectiveness);
in = input('MCM vs B (1-9): ');
AHPEffectiveness = AHPE(4,13,in,AHPEffectiveness);
```

```
in = input('ASW vs II (1-9): ');
AHPEffectiveness = AHPE(5,6,in,AHPEffectiveness);
in = input('ASW vs PD (1-9): ');
AHPEffectiveness = AHPE(5,7,in,AHPEffectiveness);
in = input('ASW vs IO (1-9): ');
AHPEffectiveness = AHPE(5,8,in,AHPEffectiveness);
in = input('ASW vs TCS (1-9): ');
AHPEffectiveness = AHPE(5,9,in,AHPEffectiveness);
in = input('ASW vs Vs (1-9): ');
AHPEffectiveness = AHPE(5,10,in,AHPEffectiveness);
in = input('ASW vs Ve (1-9): ');
AHPEffectiveness = AHPE(5,11,in,AHPEffectiveness);
in = input('ASW vs Ed (1-9): ');
AHPEffectiveness = AHPE(5,12,in,AHPEffectiveness);
in = input('ASW vs B (1-9): ');
AHPEffectiveness = AHPE(5,13,in,AHPEffectiveness);

in = input('II vs PD (1-9): ');
AHPEffectiveness = AHPE(6,7,in,AHPEffectiveness);
in = input('II vs IO (1-9): ');
AHPEffectiveness = AHPE(6,8,in,AHPEffectiveness);
in = input('II vs TCS (1-9): ');
AHPEffectiveness = AHPE(6,9,in,AHPEffectiveness);
in = input('II vs Vs (1-9): ');
AHPEffectiveness = AHPE(6,10,in,AHPEffectiveness);
in = input('II vs Ve (1-9): ');
AHPEffectiveness = AHPE(6,11,in,AHPEffectiveness);
in = input('II vs Ed (1-9): ');
AHPEffectiveness = AHPE(6,12,in,AHPEffectiveness);
in = input('II vs B (1-9): ');
AHPEffectiveness = AHPE(6,13,in,AHPEffectiveness);

in = input('PD vs IO (1-9): ');
AHPEffectiveness = AHPE(7,8,in,AHPEffectiveness);
in = input('PD vs TCS (1-9): ');
AHPEffectiveness = AHPE(7,9,in,AHPEffectiveness);
in = input('PD vs Vs (1-9): ');
AHPEffectiveness = AHPE(7,10,in,AHPEffectiveness);
in = input('PD vs Ve (1-9): ');
AHPEffectiveness = AHPE(7,11,in,AHPEffectiveness);
in = input('PD vs Ed (1-9): ');
AHPEffectiveness = AHPE(7,12,in,AHPEffectiveness);
in = input('PD vs B (1-9): ');
AHPEffectiveness = AHPE(7,13,in,AHPEffectiveness);

in = input('IO vs TCS (1-9): ');
AHPEffectiveness = AHPE(8,9,in,AHPEffectiveness);
```

```
in = input('IO vs Vs (1-9): ');
AHPEffectiveness = AHPE(8,10,in,AHPEffectiveness);
in = input('IO vs Ve (1-9): ');
AHPEffectiveness = AHPE(8,11,in,AHPEffectiveness);
in = input('IO vs Ed (1-9): ');
AHPEffectiveness = AHPE(8,12,in,AHPEffectiveness);
in = input('IO vs B (1-9): ');
AHPEffectiveness = AHPE(8,13,in,AHPEffectiveness);

in = input('TCS vs Vs (1-9): ');
AHPEffectiveness = AHPE(9,10,in,AHPEffectiveness);
in = input('TCS vs Ve (1-9): ');
AHPEffectiveness = AHPE(9,11,in,AHPEffectiveness);
in = input('TCS vs Ed (1-9): ');
AHPEffectiveness = AHPE(9,12,in,AHPEffectiveness);
in = input('TCS vs B (1-9): ');
AHPEffectiveness = AHPE(9,13,in,AHPEffectiveness);

in = input('Vs vs Ve (1-9): ');
AHPEffectiveness = AHPE(10,11,in,AHPEffectiveness);
in = input('Vs vs Ed (1-9): ');
AHPEffectiveness = AHPE(10,12,in,AHPEffectiveness);
in = input('Vs vs B (1-9): ');
AHPEffectiveness = AHPE(10,13,in,AHPEffectiveness);

in = input('Ve vs Ed (1-9): ');
AHPEffectiveness = AHPE(11,12,in,AHPEffectiveness);
in = input('Ve vs B (1-9): ');
AHPEffectiveness = AHPE(11,13,in,AHPEffectiveness);

in = input('Ed vs B (1-9): ');
AHPEffectiveness = AHPE(12,13,in,AHPEffectiveness);

%% Calculate Weights
for j = 1:1:13
    AHPEffectiveness(:,j) =
AHPEffectiveness(:,j)/sum(AHPEffectiveness(:,j));
end

for i = 1:1:13
    disp(['EW',num2str(i),' =
',num2str(mean(AHPEffectiveness(i,:)))]);
end

%% Value function
function AHPEout = AHPE(i,j,inval,AHPEin)
AHPEout = AHPEin;
if inval == 1
```

```
    AHPEout(i,j) = 9;
    AHPEout(j,i) = 1/9;
elseif inval == 2
    AHPEout(i,j) = 7;
    AHPEout(j,i) = 1/7;
elseif inval == 3
    AHPEout(i,j) = 5;
    AHPEout(j,i) = 1/5;
elseif inval == 4
    AHPEout(i,j) = 3;
    AHPEout(j,i) = 1/3;
elseif inval == 5
    AHPEout(i,j) = 1;
    AHPEout(j,i) = 1;
elseif inval == 6
    AHPEout(i,j) = 1/3;
    AHPEout(j,i) = 3;
elseif inval == 7
    AHPEout(i,j) = 1/5;
    AHPEout(j,i) = 5;
elseif inval == 8
    AHPEout(i,j) = 1/7;
    AHPEout(j,i) = 7;
elseif inval == 9
    AHPEout(i,j) = 1/9;
    AHPEout(j,i) = 9;
end
```

B. AHP Risk Code

```
function AHPRtest()
%% Analytical Heirarchy Process Script

AHPRisk = ones(3);
in = input('Performance vs Schedule (1-9): ');
AHPRisk = AHPE(1,2,in,AHPRisk);
in = input('Performance vs Cost (1-9): ');
AHPRisk = AHPE(1,3,in,AHPRisk);

in = input('Schedule vs Cost (1-9): ');
AHPRisk = AHPE(2,3,in,AHPRisk);

%% Calculate Weights
for j = 1:1:3
    AHPRisk(:,j) = AHPRisk(:,j)/sum(AHPRisk(:,j));
end

for i = 1:1:3
```

```
    disp(['RW1 = ',num2str(mean(AHPRisk(i,:)))]);
end


%% Value function
function AHPEout = AHPE(i,j,inval,AHPEin)
AHPEout = AHPEin;
if inval == 1
    AHPEout(i,j) = 9;
    AHPEout(j,i) = 1/9;
elseif inval == 2
    AHPEout(i,j) = 7;
    AHPEout(j,i) = 1/7;
elseif inval == 3
    AHPEout(i,j) = 5;
    AHPEout(j,i) = 1/5;
elseif inval == 4
    AHPEout(i,j) = 3;
    AHPEout(j,i) = 1/3;
elseif inval == 5
    AHPEout(i,j) = 1;
    AHPEout(j,i) = 1;
elseif inval == 6
    AHPEout(i,j) = 1/3;
    AHPEout(j,i) = 3;
elseif inval == 7
    AHPEout(i,j) = 1/5;
    AHPEout(j,i) = 5;
elseif inval == 8
    AHPEout(i,j) = 1/7;
    AHPEout(j,i) = 7;
elseif inval == 9
    AHPEout(i,j) = 1/9;
    AHPEout(j,i) = 9;
end
```

C. Electronics Code

```
function [PowAvail,PowReqd,F_PR] =
AUVElectronics2(PayConf,PropConf,ElecConf,BatConf,CommConf)

%% Import Configuration File Config.xls
newData1 = importdata('Config.txt');
CompNumAll = newData1.data(:,3);
ConfigAll = newData1.data(:,2);
ConfigTypeAll = newData1.data(:,1);

clear newData1
```

```
%% Make list of components needed
PayInd = find(ConfigTypeAll == 1);
TempConfig = ConfigAll(PayInd);
TempCompNum = CompNumAll(PayInd);
clear PayInd
PayInd = find(TempConfig == PayConf);
CompNum = TempCompNum(PayInd);
clear PayInd TempConfig TempCompNum

PropInd = find(ConfigTypeAll == 2);
TempConfig = ConfigAll(PropInd);
TempCompNum = CompNumAll(PropInd);
clear PropInd
PropInd = find(TempConfig == PropConf);
CompNum = [CompNum; TempCompNum(PropInd);];
clear PropInd TempConfig TempCompNum

ElecInd = find(ConfigTypeAll == 3);
TempConfig = ConfigAll(ElecInd);
TempCompNum = CompNumAll(ElecInd);
clear ElecInd
ElecInd = find(TempConfig == ElecConf);
CompNum = [CompNum; TempCompNum(ElecInd);];
clear ElecInd TempConfig TempCompNum

BatInd = find(ConfigTypeAll == 4);
TempConfig = ConfigAll(BatInd);
TempCompNum = CompNumAll(BatInd);
clear BatInd
BatInd = find(TempConfig == BatConf);
CompNum = [CompNum; TempCompNum(BatInd);];
clear ElecInd TempConfig TempCompNum

CommInd = find(ConfigTypeAll == 5);
TempConfig = ConfigAll(CommInd);
TempCompNum = CompNumAll(CommInd);
clear CommInd
CommInd = find(TempConfig == CommConf);
CompNum = [CompNum; TempCompNum(CommInd);];
clear CommInd TempConfig TempCompNum CompNumAll ConfigAll
ConfigTypeAll i


%% Import Components from Input.xls
newData1 = importdata('Input.txt');

x = newData1.data(:,2);
```

```
x1 = newData1.data(:,3);
x2 = newData1.data(:,4);
x3 = newData1.data(:,5:15);

clear newData1

%% Make a list of what you need
ind = find(x == CompNum(1));
VoltageRequired = x1(ind);
PowerUsed = x2(ind);
Port = x3(ind,:);


for j = 2:1:length(CompNum)
    clear ind
    ind = find(x == CompNum(j));
    VoltageRequired = [VoltageRequired; x1(ind);];
    PowerUsed = [PowerUsed; x2(ind);];
    Port = [Port; x3(ind,:);];
end

clear x x1 x2 x3 ind

%% Calculate
VoltAvail,VoltReqd,PowAvail,PowReqd,PortAvail,PortReqd
CurrentReqd = 0;
CurrentAvail = 0;
VoltReqd = 0;
VoltAvail = 0;
PortAvail = zeros(1,11);
PortReqd = zeros(1,11);
for k = 1:1:length(VoltageRequired)
    for j = 1:1:11
        if Port(k,j) > 0
            PortAvail(j) = PortAvail(j) + Port(k,j);
        elseif Port(k,j) < 0
            PortReqd(j) = PortReqd(j) + abs(Port(k,j));
        end
    end
    check = 0;
    for z = 1:1:length(VoltReqd)
        if VoltReqd(z) == VoltageRequired(k)
            check = 1;
        end
    end
    for y = 1:1:length(VoltAvail)
        if VoltAvail(y) == VoltageRequired(k)
            check = 1;
```

```
        end
    end
    if check == 0 && VoltageRequired(k) < 0
        VoltReqd = [VoltReqd; VoltageRequired(k)];
        CurrentReqd = [CurrentReqd; PowerUsed(k)];
    elseif check == 0 && VoltageRequired(k) > 0
        VoltAvail = [VoltAvail; VoltageRequired(k)];
        CurrentAvail = [CurrentAvail; PowerUsed(k)];
    end
end
TempVA = VoltAvail(2:length(VoltAvail));
TempVR = -VoltReqd(2:length(VoltReqd));
TempCA = CurrentAvail(2:length(CurrentAvail));
TempCR = CurrentReqd(2:length(CurrentReqd));
clear VoltAvail VoltReqd CurrentAvail CurrentReqd
VoltReqd = TempVR;
VoltAvail = TempVA;
CurrentReqd = TempCR;
CurrentAvail = TempCA;
clear TempVA TempVR TempCR TempCA

%Calculate Power Stuff next
PowAvail = sum(VoltAvail.*CurrentAvail); %See note below
PowReqd = sum(VoltReqd.*CurrentReqd); %See note below

%% Calculate Port Feasibility
F_PR = (sum(PortAvail)-sum(PortReqd))/sum(PortReqd);
```

D. Hull Code

```
function
[Weight,Ballast,CGX,CGY,CGZ,Clength,MidVolAvail,TotalVolAva
il,Lmid,HullWeight,MidWeight,CapWeight] =
AUVHull2(D,LtoD,BtoD,na,nf,HullMat,WallT,PayConf,PropConf,E
lecConf,BatConf,CommConf)

%% Import Configuration File Config.xls
newData1 = importdata('Config.txt');
CompNumAll = newData1.data(:,3);
ConfigAll = newData1.data(:,2);
ConfigTypeAll = newData1.data(:,1);

clear newData1

%% Make list of components needed
PayInd = find(ConfigTypeAll == 1);
TempConfig = ConfigAll(PayInd);
TempCompNum = CompNumAll(PayInd);
```

```
clear PayInd
PayInd = find(TempConfig == PayConf);
CompNum = TempCompNum(PayInd);
clear PayInd TempConfig TempCompNum

PropInd = find(ConfigTypeAll == 2);
TempConfig = ConfigAll(PropInd);
TempCompNum = CompNumAll(PropInd);
clear PropInd
PropInd = find(TempConfig == PropConf);
CompNum = [CompNum; TempCompNum(PropInd);];
clear PropInd TempConfig TempCompNum

ElecInd = find(ConfigTypeAll == 3);
TempConfig = ConfigAll(ElecInd);
TempCompNum = CompNumAll(ElecInd);
clear ElecInd
ElecInd = find(TempConfig == ElecConf);
CompNum = [CompNum; TempCompNum(ElecInd);];
clear ElecInd TempConfig TempCompNum

BatInd = find(ConfigTypeAll == 4);
TempConfig = ConfigAll(BatInd);
TempCompNum = CompNumAll(BatInd);
clear BatInd
BatInd = find(TempConfig == BatConf);
CompNum = [CompNum; TempCompNum(BatInd);];
clear ElecInd TempConfig TempCompNum

CommInd = find(ConfigTypeAll == 5);
TempConfig = ConfigAll(CommInd);
TempCompNum = CompNumAll(CommInd);
clear CommInd
CommInd = find(TempConfig == CommConf);
CompNum = [CompNum; TempCompNum(CommInd);];
clear CommInd TempConfig TempCompNum CompNumAll ConfigAll
ConfigTypeAll i

%% Import Components from Input.xls
newData1 = importdata('Input.txt');
x = newData1.data(:,2);
x16 = newData1.data(:,17);
x17 = newData1.data(:,18);
x18 = newData1.data(:,19);
x19 = newData1.data(:,20);
x20 = newData1.data(:,21);
x21 = newData1.data(:,22);
x22 = newData1.data(:,23);
```

```
x15 = newData1.data(:,16);

clear newData1

%% Make a list of what you need
ind = find(x == CompNum(1));
X = x16(ind);
Y = x17(ind);
Z = x18(ind);
XCG = x19(ind);
YCG = x20(ind);
ZCG = x21(ind);
Loc = x22(ind);
W = x15(ind);


for j = 2:1:length(CompNum)
    clear ind
    ind = find(x == CompNum(j));
    X = [X; x16(ind);];
    Y = [Y; x17(ind);];
    Z = [Z; x18(ind);];
    XCG = [XCG; x19(ind);];
    YCG = [YCG; x20(ind);];
    ZCG = [ZCG; x21(ind);];
    Loc = [Loc; x22(ind);];
    W = [W; x15(ind);];
end

clear x x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15
x16 x17 x18 x19 x20
clear x21 x22 x23 x24 x25 ind

X = X/1000;
Y = Y/1000;
Z = Z/1000;
XCG = XCG/1000;
YCG = YCG/1000;
ZCG = ZCG/1000;
W = W/1000;

%Make X > Y > Z
for j = 1:1:length(X);
    clear temp tempcg;
    if X(j) >= Y(j) && Y(j) >= Z(j)
        %nothing
    elseif X(j) >= Y(j) && Z(j) >= Y(j)
        temp = Y(j);
```

```
            Y(j) = Z(j);
            Z(j) = temp;
            tempcg = YCG(j);
            YCG(j) = ZCG(j);
            ZCG(j) = tempcg;
        elseif Y(j) >= X(j) && Y(j) >= Z(j)
            if X(j) >= Z(j)
                temp = X(j);
                X(j) = Y(j);
                Y(j) = temp;
                tempcg = XCG(j);
                XCG(j) = YCG(j);
                YCG(j) = tempcg;
            else
                temp = Z(j);
                Z(j) = X(j);
                X(j) = Y(j);
                Y(j) = temp;
                tempcg = ZCG(j);
                ZCG(j) = XCG(j);
                XCG(j) = YCG(j);
                YCG(j) = tempcg;
            end
        elseif Y(j) >= X(j) && Z(j) >= Y(j)
            temp = X(j);
            X(j) = Z(j);
            Z(j) = X(j);
            Y(j) = temp;
            tempcg = XCG(j);
            XCG(j) = ZCG(j);
            ZCG(j) = XCG(j);
            YCG(j) = tempcg;
        end
    end
    clear temp tempcg

    %Only arrange cylinder objects (location 2)
    clear ind
    ind = find(Loc == 2);
    Xtemp = X(ind);
    Ytemp = Y(ind);
    Ztemp = Z(ind);
    XCGtemp = XCG(ind);
    YCGtemp = YCG(ind);
    ZCGtemp = ZCG(ind);
    Wtemp = W(ind);
    %Still need weight for nose and tail sections
    clear ind
```

```
ind = find(Loc == 1);
Wnose = sum(W(ind)); %nose weight (not including nose
materials)
Volnose = sum(X(ind).*Y(ind).*Z(ind));
clear ind
ind = find(Loc == 3);
Wtail = sum(W(ind)); %tail weight (not including tail
materials)
Voltail = sum(X(ind).*Y(ind).*Z(ind));
clear X Y Z XCG YCG ZCG W

%Sort by Weight
[temp,ind] = sort(Wtemp,'descend');
X = Xtemp(ind);
Y = Ytemp(ind);
Z = Ztemp(ind);
XCG = XCGtemp(ind);
YCG = YCGtemp(ind);
ZCG = ZCGtemp(ind);
W = Wtemp(ind);
clear Xtemp Ytemp Ztemp XCGtemp YCGtemp ZCGtemp Wtemp ind

%% Arrange
N = length(X);

R = D/2;
RB = BtoD*D/2;
ArrangeX = ones(length(X),1);
ArrangeY = ones(length(X),1);
ArrangeZ = ones(length(X),1);

% Generate boxes, used for inpolygon checking
for j = 1:1:N
    xbox(:,j) = ones(105,1)*X(j);
    ybox(:,j) = [transpose(Y(j)/2:-Y(j)/50:-Y(j)/2); -
Y(j)/2; transpose(-Y(j)/2:Y(j)/50:Y(j)/2);   Y(j)/2;
Y(j)/2;];
    zbox(:,j) = [ones(51,1)*Z(j)/2;  0;       ones(51,1)*-
Z(j)/2;  0;         Z(j)/2;];
end

unused = (1:1:N);
used = 0;
xib = (1:1:N)*0;
xie = xib;
xi = 0;

while sum(unused) ~= 0
```

```
    for j = 1:1:length(unused)
        fail = 0;
        if unused(j) == 0
        else
            yi = Y(unused(j))/2 -
R*sin(acos((Z(unused(j))/2)/RB));  %places as low as
possible on the circle
            for m = 1:1:length(used)
                if fail == 0
                    if used(m) == 0
                    else
                        [IN,ON] =
inpolygon(ybox(:,unused(j))+yi,zbox(:,unused(j)),ybox(:,use
d(m)),zbox(:,used(m)));
                        if sum(IN)-sum(ON) == 0  %Outside
of other boxes while at bottom of circle?
                        else
                            fail = 1;
                        end
                    end
                end
            end
            if fail == 1
                %fails test, move box
                pass = 0;
                for m = 1:1:length(used) %m indicates where
it's been moved to
                    fail2 = 0;
                    if pass == 0
                        if used(m) == 0 %has m been used
before? 0 if no
                        else
                            yi = max(ybox(:,used(m))) +
Y(unused(j))/2; %put new box ontop of box m
                            INf1 =
inpolygon(ybox(:,unused(j))+yi,zbox(:,unused(j)),RB*cos(0:.
01:2*pi),R*sin(0:.01:2*pi));
                            if sum(INf1) == length(INf1)
%is it still inside the circle?
                                for n = 1:1:length(used)
%loop to check overlapping boxes
                                    if fail2 == 0
                                        if used(n) == 0
                                        else
                                            [INf2,ONf2] =
inpolygon(ybox(:,unused(j))+yi,zbox(:,unused(j)),ybox(:,use
d(n)),zbox(:,used(n)));
                                            if sum(INf2)-
```

```
sum(ONf2) == 0
                                                    else
                                                        fail2 = 1;
                                                    end
                                                    [INf3,ONf3] =
inpolygon(ybox(:,used(n)),zbox(:,used(n)),ybox(:,unused(j))
+yi,zbox(:,unused(j)));
                                                    if sum(INf3)-
sum(ONf3) == 0
                                                    else
                                                        fail2 = 1;
                                                    end
                                                end
                                            end
                                        end
                                        if fail2 == 1
                                        else
                                            pass = 1;
                                            ybox(:,unused(j)) =
ybox(:,unused(j)) + yi;

used = [used unused(j)];
                                            xib(unused(j)) = xi;
                                            xie(unused(j)) =
xi+X(unused(j));
                                            ArrangeX(unused(j)) =
xi+XCG(unused(j));
                                            ArrangeY(unused(j)) =
min(ybox(:,unused(j)))+YCG(unused(j));
                                            ArrangeZ(unused(j)) =
min(zbox(:,unused(j)))+ZCG(unused(j));
                                            unused(j) = 0;
                                        end
                                    end
                                end
                            end
                        end
                    else
                        %passes tests, place box
                        ybox(:,unused(j)) = ybox(:,unused(j)) + yi;
used = [used unused(j)];
                        xib(unused(j)) = xi;
                        xie(unused(j)) = xi+X(unused(j));
                        ArrangeX(unused(j)) = xi+XCG(unused(j));
                        ArrangeY(unused(j)) =
min(ybox(:,unused(j)))+YCG(unused(j));
                        ArrangeZ(unused(j)) =
min(zbox(:,unused(j)))+ZCG(unused(j));
```

```
                unused(j) = 0;
            end
        end
    end
    if sum(unused) == 0
        break;
    end
    countunused = 1;
    countused = 1;
    for j = 1:1:length(unused)
        if unused(j) == 0
        else
            unusedtemp(countunused) = unused(j);
            countunused = countunused+1;
        end
    end
    if used(1) == 0 && length(used) == 1
        [minx,minind] = max(xie);
    elseif used(1) == 0
        [minx,minind] = min(xie(used(2:1:length(used))));
        minind = used(1+minind);
    else
        [minx,minind] = min(xie(used));
        minind = used(minind);
    end
    for j = 1:1:length(used)
        if length(used) == 1
            usedtemp = 0;
        elseif length(used) == 2 && sum(used) == minind
            usedtemp = 0;
        elseif used(j) == 0
        elseif used(j) == minind
        else
            usedtemp(countused) = used(j);
            countused = countused+1;
        end
        xi = xie(minind);
    end
    clear unused used
    unused = unusedtemp;
    used = usedtemp;
    clear unusedtemp usedtemp countused countunused
end

%% Hull Characteristics (from Senior Design)
LOA=LtoD*D;
B=BtoD*D;
Lmid=.5*LOA;
```

```
del=B-D;
Lfwd=.25*LOA;
Laft=.25*LOA;
Vf1 = qsimp2([1,Lfwd,Lmid,na,nf,D,Laft],0.,Lfwd);
Va1 = qsimp2([2,Lfwd,Lmid,na,nf,D,Laft],Lfwd+Lmid,LOA);
Vf2 = qsimp2([3,Lfwd,Lmid,na,nf,D,Laft],0.,Lfwd);
Va2 = qsimp2([4,Lfwd,Lmid,na,nf,D,Laft],Lfwd+Lmid,LOA);
Sf1=2*pi*Vf2;
Sa1=2*pi*Va2;
Vf2=2*Vf2*del;
Va2=2*Va2*del;
S=Sf1+2*pi*D/2.*Lmid+Sa1+2.*del*1.05*LOA;
% total surface area
Venv=Vf1+Lmid*pi*(D/2.)^2+Va1+Vf2+D*del*Lmid+Va2;
% total envelope volume

%% Calculate Ballast
% Calculate total volume
CompVol = X.*Y.*Z;
MidCompVol = sum(CompVol);
TotalCompVol = MidCompVol + Volnose + Voltail;
BodyVol = Lmid*pi*R^2;
TotalBodyVol = Venv;

% Calculate total weight
TotalWeight = sum(W);

% Calculate displacement based on density of freshwater
rho_water = 1000; %kg/m^3, need to determine what units to
use
TotalBuoy = BodyVol*rho_water;

%Hull weight
MidWeight = HullMat*WallT*2*pi*R*Lmid;
CapWeight = 2700*WallT*(S-2*pi*R*Lmid);
HullWeight = MidWeight + CapWeight;

% Calculate Ballast Weight
Ballast = TotalBuoy - TotalWeight - HullWeight;

MidVolAvail = BodyVol-MidCompVol;
TotalVolAvail = TotalBodyVol-TotalCompVol;

%     %Output Arrengment to Data File
%     %Close File
% end
% cd ..
```

```
%% Calculate CG
%Assume CGY and CGZ for Nose/Tail are 0
%Need W from non-body sections
%Ballast not included in CG calculations
noseoffset = .2*LOA;
tailoffset = .8*LOA;
ArrangeX = ArrangeX + .25*LOA;
Weight = sum(W) + Wnose + Wtail;
CGX = sum(ArrangeX.*W + Wnose*noseoffset +
Wtail*tailoffset)/(Weight);
CGY = sum(ArrangeY.*W)/(Weight);
CGZ = sum(ArrangeZ.*W)/(Weight);

%% Compressed length
Clength = max(xie);
```

E. Resistance Code

```
function [Vsprint,EDuration,ERange] =
AUVResistance2(D,LtoD,BtoD,na,nf,Ve,PC,eta,PayConf,PropConf
,ElecConf,BatConf,CommConf,PowReqd,PowAvail)

%% Import Configuration File Config.xls
newData1 = importdata('Config.txt');
CompNumAll = newData1.data(:,3);
ConfigAll = newData1.data(:,2);
ConfigTypeAll = newData1.data(:,1);

clear newData1

%% Make list of components needed
PayInd = find(ConfigTypeAll == 1);
TempConfig = ConfigAll(PayInd);
TempCompNum = CompNumAll(PayInd);
clear PayInd
PayInd = find(TempConfig == PayConf);
CompNum = TempCompNum(PayInd);
clear PayInd TempConfig TempCompNum

PropInd = find(ConfigTypeAll == 2);
TempConfig = ConfigAll(PropInd);
TempCompNum = CompNumAll(PropInd);
clear PropInd
PropInd = find(TempConfig == PropConf);
CompNum = [CompNum; TempCompNum(PropInd);];
clear PropInd TempConfig TempCompNum
```

```
ElecInd = find(ConfigTypeAll == 3);
TempConfig = ConfigAll(ElecInd);
TempCompNum = CompNumAll(ElecInd);
clear ElecInd
ElecInd = find(TempConfig == ElecConf);
CompNum = [CompNum; TempCompNum(ElecInd);];
clear ElecInd TempConfig TempCompNum

BatInd = find(ConfigTypeAll == 4);
TempConfig = ConfigAll(BatInd);
TempCompNum = CompNumAll(BatInd);
clear BatInd
BatInd = find(TempConfig == BatConf);
CompNum = [CompNum; TempCompNum(BatInd);];
clear ElecInd TempConfig TempCompNum

CommInd = find(ConfigTypeAll == 5);
TempConfig = ConfigAll(CommInd);
TempCompNum = CompNumAll(CommInd);
clear CommInd
CommInd = find(TempConfig == CommConf);
CompNum = [CompNum; TempCompNum(CommInd);];
clear CommInd TempConfig TempCompNum CompNumAll ConfigAll
ConfigTypeAll i

%% Import Components from Input.xls
newData1 = importdata('Input.txt');
x = newData1.data(:,2);
x24 = newData1.data(:,25);
x23 = newData1.data(:,24);

%% Make a list of what you need
ind = find(x == CompNum(1));
AddDrag = x24(ind); %AddDrag corresponds to C_V_AP*S_AP
which is the
                    %Viscous Drag Coefficient of the
Appendage times
                    %the Surface area of the appendage
PropOut = x23(ind);


for j = 2:1:length(CompNum)
    clear ind
    ind = find(x == CompNum(j));
    AddDrag = [AddDrag; x24(ind);];
    PropOut = [PropOut; x23(ind);];
end
```

```
clear x x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15
x16 x17 x18 x19 x20
clear x21 x22 x23 x24 x25 ind

%% Hull Characteristics (from Senior Design)
De = D*3.2808399;
LOA=LtoD*De;
B=BtoD*De;
Lmid=.5*LOA;
del=B-De;
Lfwd=.25*LOA;
Laft=.25*LOA;
Vf1 = qsimp2([1,Lfwd,Lmid,na,nf,De,Laft],0.,Lfwd);
Va1 = qsimp2([2,Lfwd,Lmid,na,nf,De,Laft],Lfwd+Lmid,LOA);
Vf2 = qsimp2([3,Lfwd,Lmid,na,nf,De,Laft],0.,Lfwd);
Va2 = qsimp2([4,Lfwd,Lmid,na,nf,De,Laft],Lfwd+Lmid,LOA);
Sf1=2*pi*Vf2;
Sa1=2*pi*Va2;
Vf2=2*Vf2*del;
Va2=2*Va2*del;
S=Sf1+2*pi*De/2.*Lmid+Sa1+2.*del*1.05*LOA;
% total surface area
Venv=Vf1+Lmid*pi*(De/2.)^2+Va1+Vf2+De*del*Lmid+Va2;
% total envelope volume

%% Resistance calculations (from Gillmer and Johnson)
rho = 1.9905; %density of sea water
C_A = 0.0002; %roughness allowance for full-scale
resistance estimates made w/o model tests
S_BH = S; %wetted surface area of bare hull
nu = 1.279*10^-5;
Vef = Ve*1.6878099; %speed in ft/s
Re = Vef*LtoD*De/nu;

C_F = 0.075/(log10(Re)-2)^2;
C_V_BH = C_F*(1 + .5*(1/LtoD) + 3*(1/LtoD)^(7-nf-na/2));

EHP = (.5*rho*Vef^3/550)*((C_V_BH + C_A)*S_BH +
sum(AddDrag));
SHP = EHP/PC;

PropPowReqdE = SHP*745.699872/eta; %Watts, Endurance

EDuration = PowAvail/(PropPowReqdE+PowReqd); %PowAvail
should be in W-hrs
ERange = EDuration*Ve;

motorout = sum(PropOut);
```

```
Vs = 0.01;
for V = 0.01:1:Ve+100
    Res = V*LtoD*De/nu;
    C_Fs = 0.075/(log10(Res)-2)^2;
    C_V_BHs = C_Fs*(1 + .5*(1/LtoD) + 3*(1/LtoD)^(7-nf-
na/2));
    EHPs = (.5*rho*V^3/550)*((C_V_BHs + C_A)*S_BH +
sum(AddDrag));
    SHPs = EHPs/PC;
    PreqS = SHPs*745.699872/eta;
    if PreqS <= motorout
        Vs = V;
    else
        break;
    end
end
clear V
for V = Vs:.01:Vs+1
    Res = V*LtoD*De/nu;
    C_Fs = 0.075/(log10(Res)-2)^2;
    C_V_BHs = C_Fs*(1 + .5*(1/LtoD) + 3*(1/LtoD)^(7-nf-
na/2));
    EHPs = (.5*rho*V^3/550)*((C_V_BHs + C_A)*S_BH +
sum(AddDrag));
    SHPs = EHPs/PC;
    PreqS = SHPs*745.699872/eta;
    if PreqS <= motorout
        Vsprint = V/1.6878099;
    end
end
```

F. Feasibility Code

```
function
[F_MB,F_TB,F_BW,F_SE,F_ML,F_TL,F_ED,F_VS,F_PR,F_VE] =...
    AUVFeasibility2(D,LtoD,Ve,Lmid,MinDuration,Vsmin,...
Ballast,Clength,MidVolAvail,TotalVolAvail,Vsprint,EDuration
,F_PR,Vemin,MinBallast)

F_MB = Ballast/MidVolAvail;
F_TB = Ballast/TotalVolAvail;
F_BW = (Ballast-MinBallast);
F_SE = (Vsprint-Ve)/Ve;
F_VS = (Vsprint - Vsmin)/Vsmin;
F_VE = (Ve - Vemin)/Vemin;
F_ML = Clength/Lmid;
F_TL = Clength/(LtoD*D);
F_ED = (EDuration-MinDuration)/MinDuration;
```

G. Cost Code

```
function [BCC] =
AUVCost(PayConf,PropConf,ElecConf,BatConf,CommConf,MidWeigh
t,CapWeight,HullMat,WallType)
%% Import Configuration File Config.xls
newData1 = importdata('Config.txt');
CompNumAll = newData1.data(:,3);
ConfigAll = newData1.data(:,2);
ConfigTypeAll = newData1.data(:,1);

clear newData1

%% Make list of components needed
PayInd = find(ConfigTypeAll == 1);
TempConfig = ConfigAll(PayInd);
TempCompNum = CompNumAll(PayInd);
clear PayInd
PayInd = find(TempConfig == PayConf);
CompNum = TempCompNum(PayInd);
clear PayInd TempConfig TempCompNum

PropInd = find(ConfigTypeAll == 2);
TempConfig = ConfigAll(PropInd);
TempCompNum = CompNumAll(PropInd);
clear PropInd
PropInd = find(TempConfig == PropConf);
CompNum = [CompNum; TempCompNum(PropInd);];
clear PropInd TempConfig TempCompNum

ElecInd = find(ConfigTypeAll == 3);
TempConfig = ConfigAll(ElecInd);
TempCompNum = CompNumAll(ElecInd);
clear ElecInd
ElecInd = find(TempConfig == ElecConf);
CompNum = [CompNum; TempCompNum(ElecInd);];
clear ElecInd TempConfig TempCompNum

BatInd = find(ConfigTypeAll == 4);
TempConfig = ConfigAll(BatInd);
TempCompNum = CompNumAll(BatInd);
clear BatInd
BatInd = find(TempConfig == BatConf);
CompNum = [CompNum; TempCompNum(BatInd);];
clear ElecInd TempConfig TempCompNum

CommInd = find(ConfigTypeAll == 5);
```

```
TempConfig = ConfigAll(CommInd);
TempCompNum = CompNumAll(CommInd);
clear CommInd
CommInd = find(TempConfig == CommConf);
CompNum = [CompNum; TempCompNum(CommInd);];
clear CommInd TempConfig TempCompNum CompNumAll ConfigAll
ConfigTypeAll i

%% Import Components from Input.xls
newData1 = importdata('Input.txt');
x = newData1.data(:,2);
x25 = newData1.data(:,26);

%% Make a list of what you need
ind = find(x == CompNum(1));
Cost = x25(ind);

for j = 2:1:length(CompNum)
    clear ind
    ind = find(x == CompNum(j));
    Cost = [Cost; x25(ind);];
end

clear x x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15
x16 x17 x18 x19 x20
clear x21 x22 x23 x24 x25 ind

%% Calculate Basic Cost of Construction

CapCost = CapWeight*2.952*6;
MidVol = MidWeight/HullMat;
if WallType == 1
    Aexp = 223.4;
    Cexp = -.48;
elseif WallType == 2
    Aexp = 841.2;
    Cexp = -0.39;
elseif WallType == 3
    Aexp = 1926;
    Cexp = -.39;
elseif WallType == 4
    Aexp = 826.8;
    Cexp = -.57;
end

MidCost = MidVol*Aexp*MidVol^Cexp;

BCC = sum(Cost) + CapCost + MidCost;
```

H. OMOR Code

```
function [OMOR] =
AUVOMOR(PayConf,ElecConf,PropConf,BatConf,CommConf,PR,SR,CR
)

%% Import Components from OMOE.xls
newData1 = importdata('OMOE.txt');
ConfigTypeAll = newData1.data(:,1);
ConfigAll = newData1.data(:,2);
ER1temp = newData1.data(:,13);
ER2temp = newData1.data(:,12);
ER3temp = newData1.data(:,14);

clear newData1

%% Make list of components needed
PayInd = find(ConfigTypeAll == 1);
TempConfig = ConfigAll(PayInd);
TempER1 = ER1temp(PayInd);
TempER2 = ER2temp(PayInd);
TempER3 = ER3temp(PayInd);
MSched(1) = max(TempER1);
MPerf(1) = max(TempER2);
MCost(1) = max(TempER3);
clear PayInd
PayInd = find(TempConfig == PayConf);
ER1 = TempER1(PayInd);
ER2 = TempER2(PayInd);
ER3 = TempER3(PayInd);
clear PayInd TempConfig TempER1 TempER2 TempER3

PropInd = find(ConfigTypeAll == 2);
TempConfig = ConfigAll(PropInd);
TempER1 = ER1temp(PropInd);
TempER2 = ER2temp(PropInd);
TempER3 = ER3temp(PropInd);
MSched(2) = max(TempER1);
MPerf(2) = max(TempER2);
MCost(2) = max(TempER3);
clear PropInd
PropInd = find(TempConfig == PropConf);
ER1 = [ER1; TempER1(PropInd)];
ER2 = [ER2; TempER2(PropInd)];
ER3 = [ER3; TempER3(PropInd)];
clear PropInd TempConfig TempER1 TempER2 TempER3
```

```
ElecInd = find(ConfigTypeAll == 3);
TempConfig = ConfigAll(ElecInd);
TempER1 = ER1temp(ElecInd);
TempER2 = ER2temp(ElecInd);
TempER3 = ER3temp(ElecInd);
MSched(3) = max(TempER1);
MPerf(3) = max(TempER2);
MCost(3) = max(TempER3);
clear ElecInd
ElecInd = find(TempConfig == ElecConf);
ER1 = [ER1; TempER1(ElecInd)];
ER2 = [ER2; TempER2(ElecInd)];
ER3 = [ER3; TempER3(ElecInd)];
clear ElecInd TempConfig TempER1 TempER2 TempER3

BatInd = find(ConfigTypeAll == 4);
TempConfig = ConfigAll(BatInd);
TempER1 = ER1temp(BatInd);
TempER2 = ER2temp(BatInd);
TempER3 = ER3temp(BatInd);
MSched(4) = max(TempER1);
MPerf(4) = max(TempER2);
MCost(4) = max(TempER3);
clear BatInd
BatInd = find(TempConfig == BatConf);
ER1 = [ER1; TempER1(BatInd)];
ER2 = [ER2; TempER2(BatInd)];
ER3 = [ER3; TempER3(BatInd)];
clear BatInd TempConfig TempER1 TempER2 TempER3

CommInd = find(ConfigTypeAll == 5);
TempConfig = ConfigAll(CommInd);
TempER1 = ER1temp(CommInd);
TempER2 = ER2temp(CommInd);
TempER3 = ER3temp(CommInd);
MSched(5) = max(TempER1);
MPerf(5) = max(TempER2);
MCost(5) = max(TempER3);
clear CommInd
CommInd = find(TempConfig == CommConf);
ER1 = [ER1; TempER1(CommInd)];
ER2 = [ER2; TempER2(CommInd)];
ER3 = [ER3; TempER3(CommInd)];
clear CommInd TempConfig TempER1 TempER2 TempER3

%% Calculate OMOR
```

```
%totals are the averages of the Effectiveness ratings
ScheA = sum(ER1)/sum(MSched);
PerfA = sum(ER2)/sum(MPerf);
CostA = sum(ER3)/sum(MCost);

OMOR = PR*PerfA + CR*CostA + SR*ScheA;
```

I. OMOE Code

```
function [OMOE] =
AUVOMOE(PayConf,ElecConf,PropConf,BatConf,CommConf,...
Vsprint,Vsmin,Vsgoal,Ve,Vemin,Vegoal,Eduration,MinDuration,
GoalDuration,Ballast,MinBallast,GoalBallast,...
EW1,EW2,EW3,EW4,EW5,EW6,EW7,EW8,EW9,EW10,EW11,EW12,EW13)

%% Import Configuration File Config.xls
newData1 = importdata('OMOE.txt');
ConfigTypeAll = newData1.data(:,1);
ConfigAll = newData1.data(:,2);
ER1temp = newData1.data(:,3);
ER2temp = newData1.data(:,4);
ER3temp = newData1.data(:,5);
ER4temp = newData1.data(:,6);
ER5temp = newData1.data(:,7);
ER6temp = newData1.data(:,8);
ER7temp = newData1.data(:,9);
ER8temp = newData1.data(:,10);
ER9temp = newData1.data(:,11);

%% Make list of components needed
PayInd = find(ConfigTypeAll == 1);
TempConfig = ConfigAll(PayInd);
TempER1 = ER1temp(PayInd);
TempER2 = ER2temp(PayInd);
TempER3 = ER3temp(PayInd);
TempER4 = ER4temp(PayInd);
TempER5 = ER5temp(PayInd);
TempER6 = ER6temp(PayInd);
TempER7 = ER7temp(PayInd);
TempER8 = ER8temp(PayInd);
TempER9 = ER9temp(PayInd);
mER1(1) = max(TempER1);
mER2(1) = max(TempER2);
mER3(1) = max(TempER3);
mER4(1) = max(TempER4);
mER5(1) = max(TempER5);
mER6(1) = max(TempER6);
mER7(1) = max(TempER7);
```

```
mER8(1) = max(TempER8);
mER9(1) = max(TempER9);
clear PayInd
PayInd = find(TempConfig == PayConf);
ER1 = TempER1(PayInd);
ER2 = TempER2(PayInd);
ER3 = TempER3(PayInd);
ER4 = TempER4(PayInd);
ER5 = TempER5(PayInd);
ER6 = TempER6(PayInd);
ER7 = TempER7(PayInd);
ER8 = TempER8(PayInd);
ER9 = TempER9(PayInd);
clear PayInd TempConfig TempER1 TempER2 TempER3 TempER4
TempER5 TempER6 TempER7 TempER8 TempER9

PropInd = find(ConfigTypeAll == 2);
TempConfig = ConfigAll(PropInd);
TempER1 = ER1temp(PropInd);
TempER2 = ER2temp(PropInd);
TempER3 = ER3temp(PropInd);
TempER4 = ER4temp(PropInd);
TempER5 = ER5temp(PropInd);
TempER6 = ER6temp(PropInd);
TempER7 = ER7temp(PropInd);
TempER8 = ER8temp(PropInd);
TempER9 = ER9temp(PropInd);
mER1(2) = max(TempER1);
mER2(2) = max(TempER2);
mER3(2) = max(TempER3);
mER4(2) = max(TempER4);
mER5(2) = max(TempER5);
mER6(2) = max(TempER6);
mER7(2) = max(TempER7);
mER8(2) = max(TempER8);
mER9(2) = max(TempER9);
clear PropInd
PropInd = find(TempConfig == PropConf);
ER1 = [ER1; TempER1(PropInd)];
ER2 = [ER2; TempER2(PropInd)];
ER3 = [ER3; TempER3(PropInd)];
ER4 = [ER4; TempER4(PropInd)];
ER5 = [ER5; TempER5(PropInd)];
ER6 = [ER6; TempER6(PropInd)];
ER7 = [ER7; TempER7(PropInd)];
ER8 = [ER8; TempER8(PropInd)];
ER9 = [ER9; TempER9(PropInd)];
clear PropInd TempConfig TempER1 TempER2 TempER3 TempER4
```

```
TempER5 TempER6 TempER7 TempER8 TempER9


ElecInd = find(ConfigTypeAll == 3);
TempConfig = ConfigAll(ElecInd);
TempER1 = ER1temp(ElecInd);
TempER2 = ER2temp(ElecInd);
TempER3 = ER3temp(ElecInd);
TempER4 = ER4temp(ElecInd);
TempER5 = ER5temp(ElecInd);
TempER6 = ER6temp(ElecInd);
TempER7 = ER7temp(ElecInd);
TempER8 = ER8temp(ElecInd);
TempER9 = ER9temp(ElecInd);
mER1(3) = max(TempER1);
mER2(3) = max(TempER2);
mER3(3) = max(TempER3);
mER4(3) = max(TempER4);
mER5(3) = max(TempER5);
mER6(3) = max(TempER6);
mER7(3) = max(TempER7);
mER8(3) = max(TempER8);
mER9(3) = max(TempER9);
clear ElecInd
ElecInd = find(TempConfig == ElecConf);
ER1 = [ER1; TempER1(ElecInd)];
ER2 = [ER2; TempER2(ElecInd)];
ER3 = [ER3; TempER3(ElecInd)];
ER4 = [ER4; TempER4(ElecInd)];
ER5 = [ER5; TempER5(ElecInd)];
ER6 = [ER6; TempER6(ElecInd)];
ER7 = [ER7; TempER7(ElecInd)];
ER8 = [ER8; TempER8(ElecInd)];
ER9 = [ER9; TempER9(ElecInd)];
clear ElecInd TempConfig TempER1 TempER2 TempER3 TempER4
TempER5 TempER6 TempER7 TempER8 TempER9

BatInd = find(ConfigTypeAll == 4);
TempConfig = ConfigAll(BatInd);
TempER1 = ER1temp(BatInd);
TempER2 = ER2temp(BatInd);
TempER3 = ER3temp(BatInd);
TempER4 = ER4temp(BatInd);
TempER5 = ER5temp(BatInd);
TempER6 = ER6temp(BatInd);
TempER7 = ER7temp(BatInd);
TempER8 = ER8temp(BatInd);
TempER9 = ER9temp(BatInd);
```

```
mER1(4) = max(TempER1);
mER2(4) = max(TempER2);
mER3(4) = max(TempER3);
mER4(4) = max(TempER4);
mER5(4) = max(TempER5);
mER6(4) = max(TempER6);
mER7(4) = max(TempER7);
mER8(4) = max(TempER8);
mER9(4) = max(TempER9);
clear BatInd
BatInd = find(TempConfig == BatConf);
ER1 = [ER1; TempER1(BatInd)];
ER2 = [ER2; TempER2(BatInd)];
ER3 = [ER3; TempER3(BatInd)];
ER4 = [ER4; TempER4(BatInd)];
ER5 = [ER5; TempER5(BatInd)];
ER6 = [ER6; TempER6(BatInd)];
ER7 = [ER7; TempER7(BatInd)];
ER8 = [ER8; TempER8(BatInd)];
ER9 = [ER9; TempER9(BatInd)];
clear BatInd TempConfig TempER1 TempER2 TempER3 TempER4
TempER5 TempER6 TempER7 TempER8 TempER9

CommInd = find(ConfigTypeAll == 5);
TempConfig = ConfigAll(CommInd);
TempER1 = ER1temp(CommInd);
TempER2 = ER2temp(CommInd);
TempER3 = ER3temp(CommInd);
TempER4 = ER4temp(CommInd);
TempER5 = ER5temp(CommInd);
TempER6 = ER6temp(CommInd);
TempER7 = ER7temp(CommInd);
TempER8 = ER8temp(CommInd);
TempER9 = ER9temp(CommInd);
mER1(5) = max(TempER1);
mER2(5) = max(TempER2);
mER3(5) = max(TempER3);
mER4(5) = max(TempER4);
mER5(5) = max(TempER5);
mER6(5) = max(TempER6);
mER7(5) = max(TempER7);
mER8(5) = max(TempER8);
mER9(5) = max(TempER9);
clear CommInd
CommInd = find(TempConfig == CommConf);
ER1 = [ER1; TempER1(CommInd)];
ER2 = [ER2; TempER2(CommInd)];
ER3 = [ER3; TempER3(CommInd)];
```

```
ER4 = [ER4; TempER4(CommInd)];
ER5 = [ER5; TempER5(CommInd)];
ER6 = [ER6; TempER6(CommInd)];
ER7 = [ER7; TempER7(CommInd)];
ER8 = [ER8; TempER8(CommInd)];
ER9 = [ER9; TempER9(CommInd)];
clear CommInd TempConfig TempER1 TempER2 TempER3 TempER4
TempER5 TempER6 TempER7 TempER8 TempER9

%% Calculate OMOE

%totals are the averages of the Effectiveness ratings
totER1 = sum(ER1)/sum(mER1);
totER2 = sum(ER2)/sum(mER2);
totER3 = sum(ER3)/sum(mER3);
totER4 = sum(ER4)/sum(mER4);
totER5 = sum(ER5)/sum(mER5);
totER6 = sum(ER6)/sum(mER6);
totER7 = sum(ER7)/sum(mER7);
totER8 = sum(ER8)/sum(mER8);
totER9 = sum(ER9)/sum(mER9);

%Rating section for evaluated characteristics
%Sprint Speed
if Vsprint < Vsmin
    ER10 = 0;
elseif Vsprint >= Vsmin && Vsprint < Vsgoal
    ER10 = 0.9*(Vsprint-Vsmin)/(Vsgoal-Vsmin);
elseif Vsprint >= Vsgoal
    ER10 = 1.0;
end

%Endurance Speed
if Ve < Vemin
    ER11 = 0;
elseif Ve >= Vemin && Ve < Vegoal
    ER11 = 0.9*(Ve-Vemin)/(Vegoal-Vemin);
elseif Ve >= Vegoal
    ER11 = 1.0;
end

%Endurance Duration
if Eduration < MinDuration
    ER12 = 0;
elseif Eduration >= MinDuration && Eduration < GoalDuration
    ER12 = 0.9*(Eduration-MinDuration)/(GoalDuration-
MinDuration);
elseif Eduration >= GoalDuration
```

```
    ER12 = 1.0;
end

%Excess Ballast (used for expandability)
if Ballast < MinBallast
    ER13 = 0;
elseif Ballast >= GoalBallast
    ER13 = 1.0;
elseif Ballast >= MinBallast && Ballast < GoalBallast
    ER13 = 0.9*(Ballast-MinBallast)/(GoalBallast-
MinBallast);
end

%OMOE is the weighted average of these
OMOE = totER1*EW1 + totER2*EW2 + totER3*EW3 + totER4*EW4 +
...
    totER5*EW5 + totER6*EW6 + totER7*EW7 + totER8*EW8 +
totER9*EW9 + ...
    ER10*EW10 + ER11*EW11 + ER12*EW12 + ER13*EW13;
```

J. Resistance Comparison Code

```
clear all
%% Input for both methods
De = 32;
D = De;%/3.2808399;
LtoD = 257.6/32;
BtoD = 1;
rho = 1.9905; %density of sea water
nu = 1.279*10^-5;
Ve = 5:1:26;
nf = .7;
na = .5;

%%Hull Characteristics (from Senior Design)
LOA = LtoD*De;
B = BtoD*De;
Lmid = .5*LOA;
del = B - D;
Lfwd = .25*LOA;
Laft = .25*LOA;
Vf1 = qsimp2([1,Lfwd,Lmid,na,nf,De,Laft],0.,Lfwd);
Va1 = qsimp2([2,Lfwd,Lmid,na,nf,De,Laft],Lfwd+Lmid,LOA);
Vf2 = qsimp2([3,Lfwd,Lmid,na,nf,De,Laft],0.,Lfwd);
Va2 = qsimp2([4,Lfwd,Lmid,na,nf,De,Laft],Lfwd+Lmid,LOA);
Sf1 = 2*pi*Vf2;
Sa1 = 2*pi*Va2;
Vf2 = 2*Vf2*del;
```

```
Va2 = 2*Va2*del;
S = Sf1 + 2*pi*De/2*Lmid + Sa1 + 2*del*1.05*LOA;
% total surface area
Venv = Vf1 + Lmid*pi*(De/2.)^2 + Va1 + Vf2 + De*del*Lmid +
Va2;                                % total envelope volume

%% Resistance Comparision:  VT Method
C_A = 0.0002; %roughness allowance for full-scale
resistance estimates made w/o model tests
S_BH = S; %wetted surface area of bare hull
Vef = Ve*1.6878099; %speed in ft/s
Re = Vef*LtoD*De/nu;

C_F = 0.075*(log10(Re)-2).^-2;
C_V_BH = C_F*(1 + .5*(1/LtoD) + 3*(1/LtoD)^(7-nf-na/2));
EHP = (.5*rho*Vef.^3/550).*(C_V_BH + C_A)*S_BH;

%% Resistance Comparison: MIT Method

C_p = Venv/(pi*(De/2)^2*LOA);
C_frf = 1 + 1.5*(1/LtoD)^1.5 + 7*(1/LtoD)^3 + .002*(C_p-
.6);
EHPMIT = 0.5*rho*Vef.^3.*(C_F*C_frf + C_A)*S_BH/550;

%% Plot
plot(Ve,EHP,'g-');
hold on
plot(Ve,EHPMIT,'--');
grid on
```