Naval Ship Concept Design: an Evolutionary Approach

by

Shmuel Shahak

B.S.M.E. Technion, Israel Institute of Technology, Haifa, 1992

Submitted to the Department of Ocean Engineering in
partial fulfillment of the requirements for the degree of

Master of Science in Naval Architecture and Marine Engineering

at the

Massachusetts Institute of Technology

February 1998

Signature of Author ....
Dept. of Ocean Engineering
January 15, 1997.

Certified by ....
Alan J. Brown
Prof. of Naval Architecture
Thesis Supervisor

Accepted by ....
J. K. Vandiver
Prof. of Ocean Engineering
Chairman, Departmental Graduate committee

Naval Ship Concept Design: an Evolutionary Approach

by

Shmuel Shahak

Submitted to the Department of Ocean Engineering on January 16, 1998, in partial fulfillment of the requirements for the degree of Master of Science in Naval Architecture and Marine Engineering

Abstract

The interdependency of parameters and the complexity of the ship model make it very difficult to find an optimal design, although many feasible solutions exist. In this research weight optimization of a Naval destroyer is conducted using an evolutionary strategy, a generalized concept of what is better known as a genetic algorithm. Various genetic operators are applied to a simplified ship design process, and examined. The best configuration includes non-uniform selection pressure, one-point crossover operator, non-uniform mutation operator, and Baker's selection mechanism applied to a rank-based selection filter. The configuration is selected as the optimization tool, and is suggested for ship concept design applications.

As a case study, the suggested optimization method is applied to meet the design and payload requirements of the U.S. Navy DDG51 class, an existing vessel weighing 8130 lton at full load. The algorithm found that with a different hull form and the same HM&E systems, the same mission requirements can be met with a 7290 lton ship, lighter by more than 10%. The characteristics predicted for the new design are confirmed by ASSET, the standard computer program used by the U.S. Navy for concept ship design.

Thesis Supervisor: Alan J. Brown.
Title: Professor of Naval Architecture.

God said to Noah, "The end of all flesh has come before me, for the earth is filled with robbery through them; and behold, I am about to destroy them from the earth. Make for yourself an Ark of gopher wood; make the Ark with compartments, and cover it inside and out with pitch. This is how you should make it - three hundred cubits the length of the Ark; fifty cubits its width; and thirty cubits its height. A window shall you make for the Ark, and to a cubit finish it from above. Put the entrance of the Ark in its side; make it with bottom, second and third decks".

Genesis, the decree of the Flood, chapter 6:13-16.

Acknowledgments

I could not complete this research without the help of several people, and it is my pleasure and obligation to acknowledge them.

I would like to acknowledge Prof. Captain A. J. Brown from the Department of Ocean Engineering at MIT. His ideas, guidance and instructions have shed the light for me throughout the entire research. I have used his extensive knowledge in Naval architecture many times, not only for issues associated with this thesis. Thanks, I learned a lot from you.

Thanks to Prof. Commander Mark Welsh of the Naval Academic office at the Department of Ocean Engineering, for his generosity and hospitality in the 13A project rooms and especially in the computer laboratory.

As a Lieutenant in the Israeli Navy, my ambition to study at MIT would have not accomplished without the help of the entire organization, but I have to give special thanks to I.N. Commander Dan Avidor, a boss and a friend, for his efforts to fulfill my dreams. I hope the education I have gained here will contribute to the strength of our Navy.

From the deep of my heart sincerest thanks are for my spouse and best friend Tally. The completion of this work would have not been possible without her patient, support, care, help and love. I will not forget to include our little son Yam also. You have no idea how big is your contribution to this research.

Many thanks are sent across the Atlantic to our families in far Israel. You are all close in our hearts.

Table of Contents

Acknowledgments 4

Table of Contents 5

List of Figures 8

List of Tables 10

Introduction 12

Ship Design and Modeling 12
Evolutionary Programs and Genetic Algorithms 13
Objective of the Research 15
Preview of the Following Chapters 15

1 The Concept of Evolutionary Algorithms 17

1.1 The Basic Idea 18
1.2 Exploitation and Exploration Conflict 20
1.3 Chromosome Representation 21
1.4 Selection 23
1.5 Crossover and Mutation 24
1.6 Why do Evolutionary Programs Work? 26
1.6.1 The Schema *26*
1.6.2 The Schema Theorem *27*

2 DDGx Self-balanced Ship Design Model 29

2.1 Design Requirements 31
2.2 General Structure 31
2.3 Ship Characteristics 33
2.4 Main Dimensions 34
2.5 Resistance Module 35
2.5.1 Frictional Drag *35*
2.5.2 Residuary Drag *37*
2.5.3 Effective Horsepower *41*
2.5.4 Power Balance *44*

2.6 Available Space Module 44
2.7 Electrical Load Module 47
2.8 Tankage Module 48
2.8.1 Fuel Tanks *49*
2.8.2 Other Tanks *52*
2.8.3 Total Volume of Tanks *53*
2.9 Weight Module 53
2.10 Space Module 55
2.11 Initial Stability Module 57
2.12 Feasibility Constraints 58
2.12.1 Sustained Speed Constraint *58*
2.12.2 Depth Constraint *58*
2.12.3 Electrical Plant Constraint *60*
2.12.4 Deckhouse Area Constraint *60*
2.12.5 Deckhouse Volume Constraint *61*
2.12.6 Initial Stability Constraints *62*
2.13 Comparison to DDG51 63
2.14 Experimental Results 66

3 Tune Up in the Unconstrained Solution Space 68

3.1 Search Parameters 69
3.2 Solution Space 70
3.3 Classical Search 72
3.4 Rank-Based Selection and Selection Pressure 75
3.5 System Parameters Tune Up 78

4 Enhanced Evolutionary Strategies in the Unconstrained Solution Space 85

4.1 Non-Uniform Selection Pressure 86
4.2 Fine Local Tuning with Non-Uniform Mutation Operator 87
4.3 Baker's Selection Mechanism 89
4.4 High Order Crossover Operator 90
4.5 Arithmetical Crossover Operator 91
4.6 Experimental Results with Combined Methods 92
4.7 Conclusions 95

5 Optimization in the Constrained Solution Space 97

5.1 Feasible Solutions in the Unconstrained Search 97
5.2 Optimization with the Racist Model 99
5.3 Optimization with the Penalty Function 102
5.4 Final Local Optimization 109
5.5 Conclusions 110

6 ASSET Model of the Optimal Ship 112

6.1 Model Construction 112
6.2 Comparison between ASSET and DDGx Self-Balanced Model 117
6.2.1 Geometry *117*
6.2.2 Resistance *118*
6.2.3 Electrical Load *120*
6.2.4 Weights and Stability *120*
6.2.5 Space *121*

6.2.6 Seakeeping *122*
6.2.7 Comparative Naval Architecture *122*
6.3 Conclusions 124

Conclusions **126**

Future Study **128**

Bibliography **130**

Appendix A: Ship Design Nomenclature **134**

Appendix B: Evolutionary Methods Nomenclature **142**

Appendix C: DDGx payload and adjustments table **143**

Appendix D: DDGx Mathcad™ Model **146**

Appendix E: Extrapolated Residuary Drag Coefficients **166**

Appendix F: DDGx Evolutionary Fortran 90 Optimization model **173**

Appendix G: DDGx Final Local Optimization Fortran Model **196**

List of Figures

Figure 1.1. *Evolutionary program general flowchart.* 20
Figure 1.2. *Roulette wheel for a population of 30 individuals.* 24
Figure 2.1. *DDGx self-balanced model flowchart.* 32
Figure 2.2. *The chromosome.* 33
Figure 2.3. *Hull determination.* 34
Figure 2.4. *Wetted area coefficient from [18].* 36
Figure 2.5. *Extrapolation rules for Gertler reanalysis data.* 39
Figure 2.6. *ASSET Extrapolation rule for Gertler's reanalysis, $B/T=2.25$, $C_P=0.48$.* 39
Figure 2.7. *Worm factors for DDG51.* 41
Figure 2.8. *Appendages drag coefficient for twin screw ship, from [18].* 42
Figure 2.9. *Frontal area calculation.* 43
Figure 2.10. *Above water hull volume calculation.* 46
Figure 2.11. *Heel constraint for depth.* 59
Figure 2.12. *Maximal deckhouse dimensions.* 62
Figure 2.13. *Self-balanced DDGx model output, with DDG51 hull parameters.* 65
Figure 2.14. *Experimental feasibility status.* 67
Figure 3.1. *Solution space.* 68
Figure 3.2. *Classical genetic algorithm results with $P_{UP}=0.1$.* 72
Figure 3.3. *Payload fraction of a randomly initialized population.* 73
Figure 3.4. *Roulette wheel in a classical search.* 74
Figure 3.5. *Experimental reproduction in a classical selection.* 74
Figure 3.6. *Probability of survival under different selection pressures.* 76
Figure 3.7. *Experimental distribution of copied chromosomes under different selection pressures.* 77
Figure 3.8. *Center of selection in rank-based selection.* 78
Figure 3.9. *Rank- based algorithm results, $P_{UP}=0.2$, $P_S=0.94$.* 80
Figure 3.10. *Roulette wheel in a rank-based algorithm.* 80
Figure 3.11. *Experimental reproduction in a rank-based selection algorithm.* 81
Figure 3.12. *Number of duplicated chromosomes per generation, $P_{UP}=0.3$.* 82
Figure 3.13. *Self balanced DDGx unconstrained best model output* 83

Figure 3.14. *Convergence to a local maximum.* 84
Figure 4.1. *The non-uniform mutation Delta function, for different gen/gen_max ratios, b=2.* 88
Figure 4.2. *Influence of the parameter b on the Delta mutation function, gen/gen_max=0.3.* 88
Figure 4.3. *Experimental reproduction according to Baker's selection algorithm.* 89
Figure 4.4. *Saturated population at final generation.* 96
Figure 5.1. *Number of feasible individuals at unconstrained exploration.* 98
Figure 5.2. *Convergence of feasible solutions in an unconstrained exploration.* 98
Figure 5.3. *Racist optimization model flowchart.* 100
Figure 5.4. *Contradiction of the evolution principle in the racist model.* 101
Figure 5.5. *Experimental maximum and Average fitness with punishment.* 106
Figure 5.6. *Number of feasible chromosomes per generation for Figure 5.5 run.* 106
Figure 5.7. *Self balanced DDGx optimal ship characteristics.* 108
Figure 5.8. *Final population with penalty function for the run of Figure 5.5.* 109
Figure 6.1. *Optimal DDGx hull general arrangement.* 113
Figure 6.2. *Optimal vessel hull and deckhouse.* 114
Figure 6.3. *DDGx mechanical arrangement.* 115
Figure 6.4. *Design summary output for DDGx optimal ship.* 116
Figure 6.5. *Comparison of frictional, residuary and total effective horsepower.* 118
Figure 6.6. *DDGx residuary horsepower comparison between predictive methods.* 119
Figure F.1. *DDGx optimization code interface window.* 173
Figure F.2. *DDGx optimization code intermediate report.* 174
Figure F.3. *DDG_report.out sample output file.* 175
Figure F.4. *DDGx Fortran code flowchart.* 176
Figure G.1. *Hill climbing Fortran code flowchart.* 197
Figure G.2. *Example of output file of hill climbing optimization program.* 198

List of Tables

Table 2.1. *Gertler original reanalysis space domain* ... 38
Table 2.2. *Comparison of DDGx model and ASSET match runs for the DDG51* ... 65
Table 3.1. *Solution space boundaries and search resolution* ... 71
Table 3.2. *Maximal payload fraction* ... 79
Table 3.3. *Generations for convergence.* ... 79
Table 4.1. *Experimental results with non-uniform selection pressure* ... 86
Table 4.2. *Experimental results with non-uniform mutation operator, b=2.* ... 89
Table 4.3. *Experimental results with Baker's selection method.* ... 90
Table 4.4. *Experimental results with two-point crossover.* ... 91
Table 4.5. *Experimental results with arithmetical crossover.* ... 92
Table 4.6. *Experimental results with combined evolutionary operators* ... 93
Table 4.7. *Experimental convergence results with combined evolutionary operators.* ... 93
Table 4.8. *Experimental results with uniform pressure, non-uniform mutation and Baker's selection* ... 94
Table 4.9. *Comparison of all evolutionary strategies experimental results* ... 94
Table 4.10. *Optimal hulls in the unrestrained solution space* ... 96
Table 5.1. *Best feasible results obtained in the unconstrained search.* ... 99
Table 5.2. *Feasible experimental results with the racist optimization model.* ... 102
Table 5.3. *Feasible experimental results with penalty function.* ... 105
Table 5.4. *Gene steps in hill climbing optimization.* ... 110
Table 5.5. *Hill climbing experimental results, start point at 0.1253.* ... 110
Table 6.1. *DDGx data bank changed parameters.* ... 113
Table 6.2. *Geometrical comparison.* ... 117
Table 6.3. *Resistance comparison.* ... 118
Table 6.4. *Comparison of resistance results for different residuary drag predictive methods.* ... 119
Table 6.5. *Electrical load comparison.* ... 120
Table 6.6. *Weight and stability comparison.* ... 120
Table 6.7. *Space Comparison.* ... 121
Table 6.8. *Seakeeping comparison.* ... 122

Table 6.9. *Comparative naval architecture ratios for the DDGx optimal ship.* 123
Table C.1. *Payload and adjustments table.* 144
Table D.1. *Extrapolated residuary drag coefficients (x1000), C_V=0.001.* 167
Table D.2. *Extrapolated residuary drag coefficients (x1000), C_V=0.002.* 168
Table D.3. *Extrapolated residuary drag coefficients (x1000), C_V=0.003.* 169
Table D.4. *Extrapolated residuary drag coefficients (x1000), C_V=0.004.* 170
Table D.5. *Extrapolated residuary drag coefficients (x1000), C_V=0.005.* 171
Table D.6. *Extrapolated residuary drag coefficients (x1000), C_V=0.006.* 172

Introduction

Ship Design and Modeling

The design of a new ship is performed in several stages, during which the level of certainty increases, and, accordingly, the design margins decrease. Many changes and modifications are made during the period of design and construction, but the early concept design, even if is conducted with a simplified tool, always has a major impact on the final design. The concept design has an important role in the design process. Usually, the general design characteristics and dimensions of the vessel are determined in this stage. It is essential therefore that this tool gives accurate predictions.

A ship is a very complex platform, with a non-analytic hull shape and various systems, the characteristics of which influence and are influenced by all others. Even a basic simplified mathematical model consists of a large number of interdependent parameters. The dependence implies that everything affects everything else. For instance, an enlarged length increases the volume of the hull, so a larger air conditioning system is required, which in turn makes the ship heavier, reducing the speed (as the wetted area is higher), but more important, increasing the length in order to provide the floatation. Hopefully, at some stage this process converges and stops. In some cases this can lead to divergence. Naval Engineers like to illustrate the design process as a spiral, where the design begins outside and converges to the inside.

Naturally, a model needs to be as scientific as possible in order to provide reliable results. In early design stages, in which a lot of data is unknown, it is hard to apply engineering methods, which require a detailed knowledge of the problem. Instead, empirical expressions are used. These are obtained by regressive analysis of actual similar vessels. As the design proceeds, more and more details are determined, and this statistical approach is replaced by logical scientific computations.

All engineering problems are actually optimization problems. Almost any real life problem has more than one solution, and many of them have an infinite number of answers. The optimal solution is the one that, among the entire population of solutions, performs best according to some specified criterion. The problem is not just to find a solution, but to find the best one. In order to keep track of solutions, models include *independent parameters*, parameters which are set at the beginning of the evaluation, and not changed. All other variables depend on these parameters, and are strictly defined by them: a solution from the infinite solution space is uniquely represented by this set. The decision on the selection of the independent parameters is not straight forward, but the effort is worthwhile, because a systematic approach is established, and the analyzer can compare the suggested solutions and choose the fittest one. The size of the *independent vector*, the vector containing the independent parameters, determines the magnitude of the solution space, and thus it is vital to have it minimized. As its length increases, the number of combinations increases as well, and the chance to find the best individual solution becomes more difficult.

A ship is a classic example. There are infinite hull shapes that meet the requirements, and the Naval Architect strives to develop the best performing one. The best solution is often a subjective opinion, since the definition depends on the purpose of the vessel. A race boat needs to have the fastest hull; a ferry is designed as to have the biggest stowage space; a tug requires a stable body with advanced maneuverability. Sometimes there are more than one consideration; usually they contradict each other. Take a ferry for example: it should have large hold spaces, to accommodate more freight, but the faster it transports it, the better; unfortunately, as the stowage spaces are increased, the displacement gets higher and the speed decreases. A decision matrix can be constructed, which assigns weights to the items and provides an overall decision criterion. Usually, however, an important consideration is the cost of the project. Cost is typically driven by weight, and is calculated through weight-based models. Hence, it is customary to conduct a weight-based optimization.

Computerized ship design optimization using is conducted for more than 30 years. Murphy, Sabat and Taylor [1] in 1965 and Mandel and Leopold [2] in 1966 conducted economical optimization to merchant ships, using very simple design models and modified random search techniques. The computers available at that period did not allow complicated computations, and consequently the models were not accurate and required many iterations.

Evolutionary Programs and Genetic Algorithms

In today's computational environment, a lot of time is spent to solve complex problems, which traditionally were considered solvable for simplified specific cases only. Researchers direct their efforts to create numerical algorithms capable of solving multi-degree and/or multi-dimensional problems. The appearance of parallel computers has strengthened this approach. Many cases, especially in engineering applications, are optimization problems, where the desired answer provides a maximum or minimum value of an objective function. The basic methods of calculus are not applicable, because usually the function is given numerically, without an explicit expression, and is often not continuous. Engineering problems are also typically highly constrained. Therefore, instead of a rigorous mathematical procedure, the solution is executed by a computer, and takes the form of a search process. A sequential-arranged search, during which all options are checked, is generally not possible, because the number of combinations is enormous (when the problem is multi dimensional), or the time required to evaluate the objective function is long (when the problem is of high order); the majority of applications involve both. Barrett [3], for example, showed that a simple exhaustive exploration would take him 2687 years[1]. There are many techniques to explore the solution space, and a review of them can be found in [4]. Among these, *evolutionary programs* are search procedures relying on analogies to natural processes, imitating the principle of evolution: *survival of the fittest*. The most basic strategy in this field is the *genetic algorithm*. Experiments with this approach can be traced to the early 1950's, but genetic algorithms as they are known today were invented by John Holland [5] at the University of Michigan. A tremendously helpful source for starters in GA's is [6], written by Michalewicz. It provides the basics, with practical examples and illustrations, suggests a lot of approaches and has a rich list of references. [7] is another familiar, more theoretical reference.

In an evolutionary search process, a collection of potential solutions, called a generation, strive for survival. They are selected according to their fitness. Better solutions have better chances to live. The surviving solutions are then allowed to breed: they experience a sequence of transformations, which imitate real life breeding. The resulting new potential solutions, the offspring, define a new generation. By the survival of the fittest principle, this generation offers enhanced performance. After a number of similar generations, the process converges, and the best individual solution represents the optimal solution.

The elementary difference between evolutionary programs and other search methods, such *as hill climbing* [9] and *simulated annealing* [10], is that evolutionary programs maintain a population of potential solutions rather than a single solution at each step. The collection of solutions imparts a wider view to the algorithm. Therefore, it performs a multi-directional search, with improved exploration capabilities and better chances to find the global optimum, instead of a local one.

[6] presents the concept of evolutionary programs as a generalization of genetic algorithms. All natural process mimics involve genetic operations, to imitate the breeding and genetic transformations that occur in nature. In genetic algorithms a potential solution is depicted by a binary string, on which two basic genetic operations are performed, crossover and mutation. Evolutionary programs, on the other hand, may use floating point number representation, and incorporate a variety of other operators.

During the last ten years there has been a growing interest in algorithms imitating natural processes, particularly in genetic algorithms. International conferences are held

[1] Page 134

around the world, [11] for example, and a world-wide academic study is being conducted. Genetic algorithms have an important usage in artificial intelligence systems, see [12] and [13], where evolutionary processes are used to instruct the machine in the best way to handle the current problem. It tries some optional solutions, evaluates their performance (via a *fitness function*), and in some sense learns the best response. Barrett [3] actually taught his mechanical Tuna fish to swim efficiently (with minimum drag) using this approach.

It seems natural, then, to apply the search engine suggested by genetic algorithms for the optimization of a ship. All the classic conditions exist: a ship model is a multi-variable problem (the variables are the independent parameters), and of high order. The solution space is huge, and an exhaustive search is impractical.

Objective of the Research

The primary objective of the research is to show that evolutionary strategies can be used in ship concept design optimization. Once this is proven, a secondary aim is to develop an effective genetic search engine, since as will be seen, this field suggests many methods. To make the research interesting and with real implications, the U.S. Naval ship DDG51 is used as a case study. The design problem is to design a vessel that carries the same (military and other) payload, and has at least the same performance: endurance, stores period and sustained speed. A very simple objective function, minimum payload weight fraction, is used. In the actual DDG51 design, many more attributes and criteria were considered. This case study is intended primarily to demonstrate the method.

Preview of the Following Chapters

There is a growing interest in evolutionary strategies and genetic algorithms in engineering applications. However, this field is relatively unfamiliar to the majority of engineers. Therefore, besides the understandable effort a graduate student makes in his thesis, I have raised an additional goal in this paper, to expose the Naval Architecture community to genetic search techniques. For this reason, this thesis includes extensive theoretical and practical background. There is no intention to create an alternative reference book; many times the reader is referred to professional literature. Nevertheless, I do hope it will impart satisfactory comprehension and stimulate the imagination of the reader.

The thesis contains distinct but evidently (in this research) connected two issues, ship design and genetic exploration. Once the required ship design foundations have been established, the concentration is on the second issue. Chapter 2 introduces the design approach, but does not employ revolutionary ship theories. It deals with ship concept design, in which a mathematical model, incorporating classical and regressive Naval Architecture techniques, is constructed. The remaining chapters, except for the last one, deal with evolutionary strategies which use the ship models. The search process evaluates different hulls, searching for the best one.

The chapters in this thesis are logically organized, starting with the basic concept, developing the engineering representation of the problem, and then reporting the experimental results of several solution approaches. The order of the chapters is according to my progress in the research. Chapter 1 introduces the basic idea of evolutionary programs, with some traditional methods and theoretical foundations to conduct a genetic search. This is an explanatory basic description only, providing the necessary background in this field, without yet touching on ship design at all. Advanced topics are introduced later, as they are incorporated in the algorithms. Chapter 2 describes the design model and the objective function which is optimized. It also defines the set of physical constraints of the system, and thus determines the boundaries of the solution space. Initial evolutionary searches are discussed in Chapter 3. These first runs are executed as sensitivity research, to obtain the suitable search parameters, such as intrinsic probabilities. These runs are conducted without the restrictions of Chapter 2. After optimal search parameters are determined, Chapter 4 describes some advanced genetic exploration results, still in the unconstrained solution space. The best mechanism is chosen to be the final search algorithm. Chapters 5 describes two different approaches taken for the constrained search. Finally, in Chapter 6, the optimal ship is analyzed and compared to the DDG51.

Chapter 1

The Concept of Evolutionary Algorithms

Evolutionary methods are probabilistic search engines, based on the mechanics of natural selection, imitating the principle of evolution. They can be used to solve all types of problems, and not just conduct optimization searches. The principle of evolution was stated in Darwin's selection theory in the mid 1800's, actually long before the discovery of genetic mechanisms. As described in [8], he did not talk about heredity principles, but on fusion or blending inheritance, suggesting that parental qualities mix together like fluids in the offspring mechanism. The theory suffered from serious objections, and only at the beginning of the 1900's, was it proved experimentally that hereditary information is carried by *chromosomes*, and transferred from parents to their descendants. In evolution, each specie looks for beneficial adaptations to better confront its environment, and the experience or knowledge it gains during its lifetime is embodied in its chromosomes. Better parents have improved chances to survive and breed, and their enhanced inheritance is then preserved. [6] illustrates[1] this by an example of rabbits and foxes: the surviving population of rabbits procreate; statistically, more smart fast rabbits survive, so their genetic contribution makes the rabbits faster and smarter on the average (because some dumb slow rabbits also survive just due to luck, and breed) from generation to generation. Of course, foxes do so as well, otherwise they would not be able to catch the rabbits, and they would be annihilated.

Evolutionary methods are a generalization of genetic algorithms. GA's were invented by Holland [5] in 1975, and during the years they have been modified as more and more research was performed in that field. Some of the new algorithms have no connection to genetics (inversion operator [6] [7], for example), and the general name of evolutionary programs was chosen. This chapter introduces the idea of evolutionary programs (starting with the specific implementation of genetic algorithms and extending to evolutionary programs), describes its fundamentals, and finally explains why they work. It starts with some basic issues, and then describes the mechanism of each step in the

[1] Page 14

algorithm. The concentration is on the very basic concepts. Other advanced algorithms, as well as experimental results, are introduced and discussed according to their appearance, in later chapters.

1.1 The Basic Idea

Evolutionary programs and genetic algorithms adopt the vocabulary of biological genetics. A potential solution in the exploration process is called a *chromosome*, to imply that this solution has some genetic information embodied in it. A chromosome can be a string of binary numbers or a vector of floating point numbers. The first type is employed in GA's, and the second in evolutionary programs. Although [6] distinguishes between these two, defining evolutionary programs as a generalization of genetic algorithms, they are not distinguished here. Occasionally, chromosomes are also called *genotypes* or *individuals*. Note the difference in the terminology: in nature, each cell of every organism carries a certain number of chromosomes; man has 46 of them. The genetic information is then determined by all chromosomes together. Evolutionary algorithms, however, store the genetic data in a single genotype. Chromosomes are made up of *genes*; each gene represents and controls a certain feature of the specie - the potential solution in the case of genetic algorithm. Each gene has a certain location on the chromosome. In order to imitate Darwin's hypothesis of fusion inheritance in offspring, we say that individuals *breed*, to create new chromosomes. For more terminology, refer to [5], [6] and [7].

The approach is to maintain at any generation a population of potential solutions, in which each chromosome represents a solution to the problem. Each solution is evaluated via the mathematical model of the problem, to determine its respective performance. The model is a tool to compute the value of the function to be optimized. This function is referred to as the *objective function* or *fitness function*. In a maximization problem, for instance, high performance of a solution is expressed by a high value of the fitness function. Then, the fittest individuals are selected to live, while the relatively bad ones die. Since bad chromosomes can still have some good genetic material (in the terminology of the rabbits, it is possible to observe a fast but however dumb rabbit), the selection scheme is probabilistic rather than elitist: it does not chose only the best individuals, but gives better chances to good chromosomes to survive. Not only does a good solution have better odds to reproduce, it also can be selected more than once. It is not rare that several chromosomes are selected more than once. In this event we say that the individual has been copied. In this way good genetic information is not lost. An elitist selection can lock on local optimum (say, converge to fast rabbits instead of fast and smart animals), because it did not use all genetic data (which, completing the analogy, would have given it information about wisdom also). The next generation must obviously suggest some new solutions, better on the average according to selection theory. This is accomplished by *genetic operators* which transfer genetic information stored in the selected population to each subsequent generation. The selected population undergoes changes by means of *crossover* and *mutation* transformations. These operators are the artificial version of breeding. They use the selected parents as their input and output the children, the new generation. Crossover serves as an analogy to reproduction. It combines the characteristics of two randomly selected individuals to create two new solutions, by breaking the original chromosomes at a randomly chosen (identical for both parents) location, and then swapping all genes on same side of the split. If, for example, the parents consist of the four gene vectors (a_1,b_1,c_1,d_1) and (a_2,b_2,c_2,d_2), and the random swap location is the second gene, the crossover operation yields the descendants (a_1,b_1,c_2,d_2) and $(a_2,b_2,,c_1,d_1)$. Note that the process is applied probabilistically, typically with a 0.25 probability[2], so some parents are transferred to the next generation just as they are, without changes. Mutation is the artificial version of genetic mutation. It randomly changes randomly selected genes of the chromosomes. In terms of the last example, in an event that second and third sites are chosen, the resulting individuals are (a_1,b_3,c_1,d_1) and (a_2,b_2,c_3,d_2). Crossover and mutation are not related to each other, meaning that it is possible to mutate a selected-but not crossed-over individual, and vice versa. Once the next generation is established, the algorithm is repeated until convergence is reached, or alternatively, the maximum allowable number of generations is passed. During all iterations the optimal generational value is stored (in a maximization problem, the maximal value evaluated among all chromosomes). Convergence is declared in the case that *n_converge* successive generations give the same optimum value, within a domain of ε. Note that evolutionary programs involve other operators, as well as higher order crossover and mutation. These are described later.

The evolutionary process is illustrated in Figure 1.1. The first generation has no heritage to use. Thus, the creation of the population is random. Obviously, the quality of the first generation influences the process: a good population converges faster to the optimum. However, it can also lead to *premature convergence*, in which the algorithm locks on a *super-individual* and concentrates only on it. Methods to prevent such a behavior are discussed in [6] and [7]; some of them are described later. Since the algorithm is probabilistic, it is usually executed several times, and the overall best solution is then picked. Only an extremely robust evolutionary program finds the same optimum in several runs. It is customary to measure the quality of a genetic search engine by the average optimum it obtains. Since the algorithm searches in a finite discrete solution space, and due to stochastic errors in the sampling mechanism of selection, it is possible that the algorithm will obtain a better fitness in earlier generations than the final optimum it reports. The chances of this event increase as the mutation rate increases: although an algorithm has a promising individual, which is selected for the next generation, it is mutated such that the new fitness is poorer. The search engine then forgets the good data and starts again, looking for other promising chromosomes. This is a good example of the effects of the exploitation versus exploration balance problem, discussed in the following section.

The genetic operators block is placed outside the main stream of the program, to imply that it is not applied to all chromosomes. It is intentional not to break down this block into crossover and mutation, because the flowchart depicts the general procedure

[2] Thus on the average only one of four pairs of parents is crossovered. See more details later.

for any evolutionary program, not only traditional genetic algorithms; the former can include additional tailored special genetic operators.

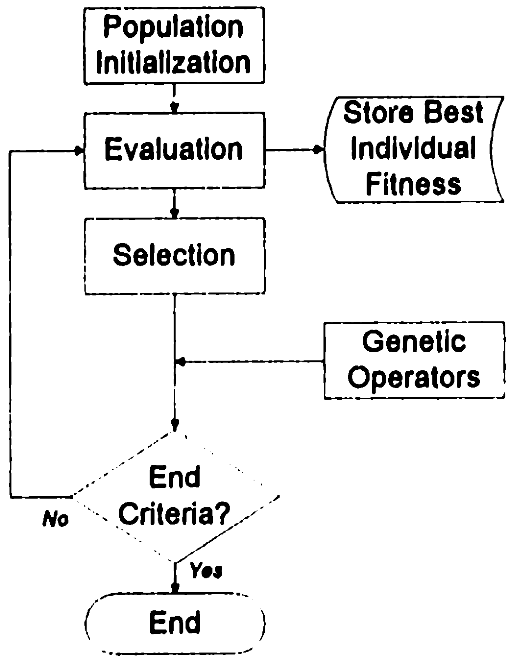


Figure 1.1. *Evolutionary program general flowchart.*

There is one essential difference between the natural evolution, as suggested by Darwin, and the evolutionary algorithms. In nature, the environment changes as well as the species, which try to enhance themselves in an effort to survive. In the allegory of the rabbits, the environmental changes are expressed by the improvement of the foxes. On the other side, in the artificial model, the objective function representing the artificial environment remains the same along the whole process. The fitness function is then said to be *static*. Dynamic environment EA's also exist, and may have applications to the ship design problem. They are not addressed in this thesis.

Any chromosome must correspond to a unique solution. The genes are therefore chosen as the independent parameters of the problem, the parameters dictating the problem which is to be solved.

1.2 Exploitation and Exploration Conflict

Genetic operators imitate the procreation process in nature, practically in that they generate the offspring, and also in the genetic sense. However, from a different point of view, without the knowledge about biological genetics, they simply provide an exploration tool, through the entire solution space. Crossover halves chromosomes to explore, while still retaining good genetic information. Mutation introduces some seemingly unnecessary accidental data, but extends the search by suggesting new solutions, still in the vicinity of relatively good options. Mutation also compensates for otherwise lost genetic data that is buried in a relatively poor individual: it introduces this data randomly in the next generation. Recall also that the initialization is random, so we must allow some variability in the process, just in case the initial population lacks important genetic material. The result is that the operators basically provide an intelligent search path, rather than just exhaustively trying random combinations of genes.

Two features dominate any search process through a space of potential solutions. First, it is expected to explore the space efficiently. Second, the algorithm should exploit the best individuals towards the optimal solution. These features oppose each other: if you want to search all over the space, you would not concentrate on a specific region; if you have found a promising region, you would not want to abandon it and search elsewhere. This issue is referred to as the *exploitation and exploration conflict*, and a search algorithm has to balance them. [6] gives two opposite examples to illustrate the applicability. Random search explores the entire solution space but ignores promising parts of it. Hill climbing [9], on the other hand, traces only the best solution for possible improvement, concentrating in a very specific region in the solution space. Evolutionary methods, however, provide a remarkable balance in this conflict: they look for the more fit individuals, but use hints from the available genetic data to explore the entire domain. This provides a multi-directional search, integrating both exploration and exploitation mechanisms.

Population diversity is a variation of the exploration feature, and is sometimes used to describe the search ability of an optimization algorithm. The larger the size of the population involved, the more genetic information is processed. The chances to lose important genetic data in the initialization are lower. The exploration is more efficient, since the algorithm has more data to learn from, and holds a better view of the solution space. Theoretically the best population size approaches infinity; however, the price is in running time, which cannot be ignored in large models.

1.3 Chromosome Representation

The chromosome consists of the independent parameters of the problem. This name is used only for complicated models, with an implicit fitness function; in a simple maximization problem of an explicit function $f(x_1, \dots, x_n)$, the arguments are the independent parameters. Actually, in an explicit problem there are no dependent parameters. Here evolutionary methods are implemented on complicated models, and this terminology is kept. The first step after the genes are defined, is to determine the solution space boundaries. Each gene gets a minimal and maximal value, and the collection of these restrictions forms the space that is explored. There are problems for which the boundaries are not known, but usually this is not the case. In ships, if the prismatic coefficient is chosen as an independent parameter, it customarily varies between 0.5 and

0.7 for naval vessels, and in any case in the domain [0,1]. The *resolution* of the search is determined as the extent to which the space is discretized. Some independent variables may be continuous, while others not. In order to have a finite (though possibly large) number of optional solutions, the solution space must be limited, allowing only discrete values for each gene. Resolution is driven by the required accuracy of the optimal solution: as the precision increases, the resolution must be increased as well. If k decimal places for the independent parameter values are required (meaning that the final optimum is expected to be accurate to k decimal digits), and if each of them takes values from a domain $D_i=[a_i,b_i]$, then the respective domain must be cut into

$$N_i = (b_i - a_i) \cdot 10^k \tag{1.1}$$

equal size ranges. The implementation is that the algorithm can have N_i+1 discrete values for the gene x_i. For instance, proceeding with the prismatic coefficient, if a reasonable precision of two decimal digits is required, $k=2$, and Equation (1.1) gives 20 sub-domains. The prismatic coefficient gene, in that case, can take 21 values (0.50, 0.51, ..., 0.70).

Two representations are possible for a chromosome. Genetic algorithms use *binary depiction*, in which the chromosome is a binary string. Each gene x_i occupies m_i bits, where m_i is the smallest integer that satisfies the inequality

$$(b_i - a_i) \cdot 10^k \leq 2^{m_i} - 1 \tag{1.2}$$

and the total string length is Σm_i. The decimal value of the gene is decoded by

$$x_i = a_i + decimal(string_2) \cdot \frac{b_i - a_i}{2^{m_i} - 1} \tag{1.3}$$

where *decimal( )* is the conversion function from binary to decimal base.

The binary representation suffers from some disadvantages, especially when utilized in intense multi-dimensional high precision models. Being a probabilistic search method, evolutionary programs employ a random number generator, operating at every generation for selection and genetic transformations processes. The generation of a random number takes time because the computer calls an external routine. Clearly, the binary depiction requires a lot of bits, because it can take only the values of 0 or 1. According to Equation (1.2), this number gets bigger as the solution space expands, and when higher precision is requested. As the length of the chromosome increases, the process takes more and more time, since a random number is generated for each bit in the genetic operations stage. Additionally, the length m_i of each gene is the smallest integer that meets Equation (1.2); it is possible to form, in the initialization or by the genetic operators, an out-of-range gene. This will not happen only if the equality in (1.2) is met. For instance, an integer gene from the domain [0,10] occupies 4 bits; it is possible for the string $(1111)_2=15_{10}$ to be obtained, which is out of the solution space. In such cases a *recovering algorithm* is written. This takes additional running time, because it requires new random number generations.

Evolutionary programs therefore use *floating number representation*. The chromosome takes the form of a vector, whose members are the genes. Its length is shorter than in binary form. The representation space is also closer to the problem space: two representatively close points are also close in their fitness performance. In binary depiction two close chromosomes can be totally remote. Floating number representation also offers accuracy for free, with no payment in chromosome length, because the computer keeps the data internally. The exploration with this representation is continuous, because the gene can have any value within its limits, and the issues of resolution discussed earlier are not of concern.

Genetic algorithms treat the chromosome logically rather than mathematically. Even if the individual has numerous genes, binary representation looks like a long string of 0's and 1's. Once the length in bits of each gene is determined, there is no longer distinction between them. In floating number representation, each gene remains separate from his peers during the whole procedure.

1.4 Selection

As the evaluation of a certain generation is completed, members of the population are selected to survive based on their relative fitness. The selection process assigns better chances to relatively better individuals. In Section 1.1 the nature of this scheme is discussed; it must allow some chance of survival for poor chromosomes also, in case they have some good genetic material. The selection scheme suggested by Holland [5] is referred to here as *classical selection*. [6] calls it the *stochastic universal sampling method*. There exist other selection methods, described in principle in [6][3], with their advantages and drawbacks. In this section only classical selection is discussed; Chapter 4 describes some other selection procedures which are used in the research.

All the selection schemes are conducted by means of a roulette wheel. This term is intentionally used, to imply that the selection is probabilistic. The wheel has number of slots equal to the population size *pop_size*, one slot per each individual. The probabilities of survival are represented by the area of the slots, sized according to the fitness of each individual, and are given by

$$P_i = \frac{fitness(x_i)}{\sum_{i=1}^{pop\_size} fitness(x_i)} \tag{1.4}$$

[3] See Section 4.1, page 56.

It can be seen that as the fitness of a certain chromosome increases, so does its probability. A typical roulette wheel is illustrated in Figure 1.2. The poorest individual gets the narrowest slot, and the best one owns the widest slot. The probabilities are sorted clockwise in a descending order, but this is not mandatory in the process.

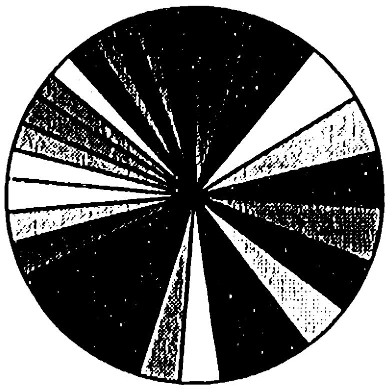


Figure 1.2. *Roulette wheel for a population of 30 individuals.*

The wheel is spun *pop_size* times, and at each spin one chromosome is selected to breed. The artificial spinning is performed with the definition of the *cumulative probability*,

$$q_i = \sum_{j=1}^{i} P_j \tag{1.5}$$

This function assigns to each individual a window, analog to the slot of the wheel, its width expresses the probability of selection, exactly as the slot does. The cumulative probability varies between the values of P_1 (for the first individual) and 1 (corresponds to the *pop_size* chromosome). The window boundaries are q_{i-1} and q_i, and the width of the i window is q_i-q_{i-1}. Instead of spinning the wheel, the computer artificially rotates the marker. It generates *pop_size* uniformly distributed random numbers between 0 and 1, playing the role of the marker: the chromosome for which the number falls in its window is selected for reproduction. The process is repeated *pop_size* times, such that pop_size individuals, not necessarily distinct, survive and continue to the next generation. Better chromosomes have a better chance to be selected more than once, in agreement with the survival of the fittest idea: better species have a better chance to preserve their heritage.

1.5 Crossover and Mutation

Some of the selected chromosomes undergo genetic transformations, and the rest are simply copied and are not changed. Genetic operators imitate natural breeding, and are the tools used to transfer genetic heredity from generation to generation. Any genetic operator is expected to direct the exploration mechanism towards a wise effective search path. This section describes only the elementary operators.

Crossover, the first operator, mates two randomly selected individuals (those who passed the selection filter) and makes them parents. In the new generation they no longer exist, but their heredity lives in their children. Depending on the genes of the parents, the offspring can be similar or totally different, poorer or hopefully performing better than the parents. The decision on whether a pair of chromosomes is crossed-over or not depends upon the *probability of crossover* P_C. This predefined parameter determines an expected value of $P_C \cdot pop\_size$ chromosomes which, on the average, undergo the crossover operation. A pair is chosen if a uniformly distributed randomly generated float number, from the domain [0,1], is less or equal to the crossover probability. The bigger the probability, the more pairs are mated to produce new offspring. After a pair is selected for crossover, a second integer random number is generated, this time from the discrete range $[1,m-1]$, where m is the total length of the chromosome (number of bits in binary representation or number of genes in floating number representation). This number indicates the position of the split, by which genetic material is swapped. [6] denotes this number by *pos*; two parents $(a_1,a_2 \ldots a_{pos},a_{pos+1} \ldots a_m)$ and $(b_1,b_2 \ldots b_{pos},b_{pos+1} \ldots b_m)$ give birth the children $(a_1,a_2 \ldots a_{pos}, b_{pos+1} \ldots b_m)$ and $(b_1,b_2 \ldots b_{pos}, a_{pos+1} \ldots a_m)$.

Mutation, the second fundamental operator, randomly changes bits of the surviving individuals. The mutation is conducted at a *mutation probability* P_M, known also as the *mutation rate*. For all existing bits, belonging to either copied or crossed-over individuals, a random float number from the domain [0,1] is produced. If it is less or equal to P_M, the bit is mutated. In binary representation, the mutation replaces a 0 by 1 and vice versa; for a float number representation, the operator replaces the gene by a new random number from the domain of that gene, $[a_i,b_i]$.

Note that the application of crossover and mutation is different for binary and floating number representation. In binary form, the chromosome is one long string, with no distinction between the various genes. The operation position can therefore occur in the middle of a gene. In the second form, all operations can take place only between genes. This is not important when dealing with crossover, because the net effect is identical. In terms of mutation, however, it can not be ignored. If the same mutation probability is used, say some high amount, binary representation will be mutated a lot more, simply due to the fact that it is much longer than the floating number form. The result in such a case is devastating: instead of a gentle mutation, intended to investigate at a near point, the offspring is thrown far away from the original location in the solution space. Similarly, if a low probability is used, the chances to mutate a floating number chromosome are too poor to be practically efficient. In order to have a uniform comparable definition, for both representations, the *probability of update* P_{UP} is introduced here. It expresses the chances that an individual is mutated at least in one bit/gene. $(1-P_M)$ is the probability that a bit (or the whole gene in floating number depiction) is not mutated; $(1\text{-}P_M)^m$ is the probability that none of the bits is mutated. The relation to the probability of mutation P_M is thus

$$P_{UP} = 1 - (1 - P_M)^m \qquad (1.6)$$

As an example, consider a two-dimensional chromosome, each gene represented by six bits in the binary form. A desired update probability of 0.3 yields, by the reversed Equation (1.6), mutation probabilities of 0.029 for the binary case and 0.163 for the floating point number depiction.

Crossover and mutation probabilities are two of the most important parameters of the search engine. The ideal balance between exploitation and exploration depends on their choice to a great extent. Each application requires a different tune of the parameters, depending on the type and behavior of the problem. Excessively high crossover probability shifts the algorithm from concentrating on high performance individuals and low probability ignores the search directions the genetic material can provide. An exaggerated mutation rate prevents convergence, because the algorithm loses the good chromosomes when reproduced for the next generation. A low mutation probability puts all the attention on exploitation while avoiding the rest of the solution space. Very often the rates are varied during the progress of the program, and the operators are classified as *dynamic*. At the beginning, the algorithm concentrates on exploration, in order to avoid premature convergence. As the process advances, the rates are decreased so the computer can search at the vicinity of only good solutions: at that stage it is believed that other portions of the space have nothing to contribute.

1.6 Why do Evolutionary Programs Work?

While the logic behind the evolutionary procedures is briefly discussed in Section 1.2, there are some theoretical foundations behind this approach. The theoretical explanation is developed in [5] for binary representation, but in conjunction with the perceptive view, it can be extended for all evolutionary representations. Basically, Holland [5] showed that fit gene groups, which he called *building blocks*, are transferred from generation to generation in exponentially increasing numbers. The optimal solution is found by joining the best building blocks together, each contributing its expertise. The exponential behavior insures rapid convergence, much better than random search. This section explains why genetic algorithms work. The same arguments with additional developments can be used for the general case of evolutionary programs.

1.6.1 The Schema

A schema is a template representing similarities among a population of chromosomes. Schemata are used to analyze the effects of genetic material transformations, performed by the genetic operators, on individuals in the populations.

Not only does this notion provide a chromosome classifying tool, it also allows investigation of the relationships between several genes in an individual.

A schema is constructed by introducing a *don't care* symbol, denoted by '*', into the binary alphabet. The don't care sign represents either '0' or '1'. Therefore, a schema represents all strings that match it in all positions other than those occupied with '*'. For example, the six bit schema (* 11000) represents two individuals, ($\underline{0}$11000) and ($\underline{1}$11000), and the schema (* 1100*) describes four strings, ($\underline{0}$1100$\underline{0}$), ($\underline{0}$1100$\underline{1}$), ($\underline{1}$1100$\underline{0}$) and ($\underline{1}$1100$\underline{1}$). Each schema matches 2^r strings, where r is the number of don't care signs in the schema. Similarly, it can be shown that from an opposite point of view, each specific individual of length m can be represented by 2^m schemata. In a population of *pop_size* optional solutions, there exist between 2^m and *pop_size*$\cdot 2^m$ different schemata. Assuming a considerably short chromosome of 30 bits ([6] describes a 3000 bit long individuals), and typical population size of 20 solutions, between 1073741824 and 32212254720 schemata may be represented.

The *order o(S)* of a schema is defined as the number of fixed positions (non '*' signs) presented in the schema. It expresses the specialty of the schema: as the order increases, less don't care signs exist, and therefore less individuals are represented by the schema. A schema of order m does not have any don't care symbols, and therefore matches only one specific chromosome. The *defining length* $\delta(S)$ is the distance between the first and last fixed positions. For instance, the schema S=(1 * * * 01 * *) has a defining length of $\delta(S)$=6-1=5. It expresses the compactness of genetic information stored in the schema. The nomenclature of [6] is adopted here. Both terms are used to compute the probabilities of survival of a schema during genetic operations.

1.6.2 The Schema Theorem

Continuing with the terminology of Michalewicz [6], the number of strings at generation t matched by a schema S is denoted by $\xi(t,S)$. The average fitness of all these ξ chromosomes is

$$F(t,S) = \frac{\sum_{i=1}^{\xi(t,S)} fitness(x_i)}{\xi(t,S)} \tag{1.7}$$

Without repetition of the entire development, it can be shown (and is shown in [6], for example) that in the next generation, the number of matched chromosomes is

$$\xi(t+1,S) \geq \xi(t,S) \cdot \frac{F(t,S)}{\overline{F}(t)} \cdot \left(1 - P_C \cdot \frac{\delta(S)}{m-1} - O(S) \cdot P_M\right) \tag{1.8}$$

Here $\bar{F}(t)$ is the generational average fitness for the entire population, including those individuals not represented by the schema S, at generation t. P_C and P_M are the probabilities of crossover and mutation, respectively. The parenthesized term is associated with the probability of survival of the schema S. This term is constant for a specific schemata and genetic system: as the order and defining length decrease, this term increases. The number of matched individuals at the next generation grows if the condition

$$\frac{F(t,S)}{\bar{F}(t)} \cdot \left(1 - P_C \cdot \frac{\delta(S)}{m-1} - O(S) \cdot P_M\right) > 1 \qquad (1.9)$$

is satisfied. This requires that the schema offers fitness $F(t,S)$ higher than the overall average generational fitness $\bar{F}(t)$. Assuming that the schema remains above average by ε%, such that $F(t,s)=(1+\varepsilon)\bar{F}(t)$, the number of matched individuals increases exponentially,

$$\xi(t,S) \geq \xi(0,S) \cdot (1+\varepsilon)^t \cdot \left(1 - P_C \cdot \frac{\delta(S)}{m-1} - O(S) \cdot P_M\right) \qquad (1.10)$$

The meaning is that a promising - above average - schema is sampled in an exponentially increasing amount. At the end, the population is saturated with individuals matched by the highest average schemata. Equation (1.10) is the mathematical representation of the schema theorem. Copied from [6], *short, low order, above average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm.* Such schemata are called the *building blocks* of the search.

The schema theorem and the terminology of order and defining length introduced before are related to the operators of crossover and mutation. Short schemata have less chance to be destroyed during crossover, and similarly low order schemata have less chance to be destroyed by the mutation operation. The combination of wise exploration offered by the genetic operators and short low order above average schemata, which exploit the best solutions, insures convergence to the optimum.

Chapter 2

DDGx Self-balanced Ship Design Model

Ship design is a tremendously complicated optimization problem. It involves a large number of parameters, and each influences the other to an unknown extent. Aside from the modeling problem, which is discussed here, there is still the philosophic argument of the definition of the optimal design. For a naval ship, it can be the fastest vessel, the least susceptible one, the one that offers the best seakeeping ability, or the one with the lightest hull. Large projects may define an integrated decision tool, which incorporates more than one consideration with respective weights to provide an overall grade for the design. The most common criterion is the cost of the project, given minimum performance constraints. Cost is traditionally modeled as a function of the weight of the vessel. The majority of cost models, particularly in preliminary design, are weight-based. This research adopts this simple approach: the best ship is considered to be the lightest one. To measure the amount of lightness, *payload weight fraction* is used, determined by

$$F_P = \frac{W_P}{W_{FL}} \tag{2.1}$$

where W_P is the weight of the military payload. The design problem is thus a maximization problem, with the payload fraction as the fitness function. The evolutionary algorithm is required to find the ship with the highest payload fraction.

This chapter describes the mathematical model used in the search process to balance the ship design variants and compute the payload fraction. The evolutionary algorithm uses this model to evaluate the fitness of each chromosome. The model is based on the MIT 13.412[1] course model, known as the *math model*. The math model is a very simple, yet effective model, developed at the MIT Ocean Engineering Department in the mid 1980's, and improved with use. It is used to demonstrate the process of designing a feasible ship, before studying the more complex U.S. Navy ship design computer program,

[1] Principles of Naval Ship Design

ASSET[2]. It employs scientific methods as well as empirical expressions, based on regression analysis of existing ships. A similar process is used in ASSET, but ASSET has many options, stores more data and provides more detailed results. In the math model, in which only one unique set of empirical expressions exists, the accuracy of the output depends on the similarity of the new design to the ships used in the regression analysis; a suitable one yields good results, comparable to ASSET. In other words, the math model is capable of giving good results for a specific class of ships. This was proven experimentally in the 13.412 third project.

The model is written in *Mathcad™ plus*, a special spreadsheet offered by *Mathsoft®*. This software has all the advantages of a regular spreadsheet, including programmability, but also allows the user to perform analysis, add graphics and annotations as with pencil and paper, using real math notation. The ability to visualize the expressions in the traditional way was very helpful during the construction of the model; this representation allows fast changes and corrections, without the need of tedious repeated compilations. The original MIT math model is also written in Mathcad. The model was called the *DDGx model*, since it is used to design ships similar to the U.S. Navy's DDG51. After the model was completed, it was converted to *Microsoft® Fortran 90* language, because Mathcad's competence is limited, in programmability, speed, problem size and complexity. The converted model is referred here as the Fortran model.

The original model was substantially changed. It is organized in a more rational order, is friendlier and more sophisticated. All required input was transferred to the head of the model, and all desired output to the end. In the old model manual input (by means of graphs and tables) is required at each iteration; in the modified model all required data is read directly, from data libraries, by the computer. The Fortran model also balances the weight and area requirements automatically, and for this reason is also called the self-balanced model. Certain algorithms demanded some corrections and modifications, usually because the MIT model was written for the design of frigates, such as FFG7. The updated DDGx Mathcad model can be used again in 13.412, and its advanced performance allows the student to spend the time on real design issues.

The most important modification is in the regression analysis. Since in this work the idea is to compare the optimization results of the genetic search to DDG51, all empirical expressions in the model are calibrated to reflect this actual ship, when its parameters are input. Although it is realized that data in the ASSET match run[3] (executed by NAVSEA) for DDG51 is not exact, the accessibility it allowed and the absence of unclassified data, together with the believed relatively high accuracy, convinced me to treat it as real. The hypothetical expressions in the model are tuned to give the same numbers. As a first choice, empirical expressions used in ASSET were taken. If they gave incorrect results, even when calibrated, the original expressions were chosen and

[2] Advanced Surface Ship Evaluation Tool

[3] Run executed for an existing vessel, in order to check and update the algorithms of the program. Some discrepancies are corrected only through P&A table.

tuned. Note that in any case, only the constants of the expressions were changed, and not the general structure. For instance, linear functions remained linear.

2.1 Design Requirements

When a new ship is to be designed, design requirements are specified by the customer or user. These demands are a set of constraints, sometimes qualitative, which the ship must satisfy. In the exploration process of this research only hull parameters (as explained in Section 2.3) are investigated. The requirements are predefined, and are treated as given.

For DDG51 comparability, its design requirements are duplicated in the DDGx model. All values are taken from the ASSET DDG51 match run. Endurance range is 3807.6 nm at 20 knots, average stores period is 45 days (ASSET distinguishes between various types of stores, and allows assignment of different values in a four-dimensional array), and average deck height (ASSET has different heights at hull and deckhouse; actually, the hull deck height is a computed number, not a given) is 10.66 feet. All explored ships have the same 922.8 lton military payload as DDG51, described in detail in Appendix C. Data for the weights, locations, area and electrical requirement is taken from ASSET payload and adjustments table. The crew is assumed to have the same size, 26 officers, 315 enlisted plus 36 additional accommodations. It is recognized that both center of gravity locations and manning size differ for two different hulls: smaller vessels have lower visual signature and smaller crew, and vice versa. In a real smaller vessel - which would hopefully be found - the payload is lower, and the crew smaller. Therefore, the calculation of stability and manning spaces is conservative. In reality the stability would be better and the required area smaller.

The HM&E[4] system is predefined for all runs, identical to DDG51. Deckhouse and hull are made of steel, with no roll fins. Four standard LM2500 gas turbines drive two controllable pitch propellers through mechanical transmission. The electrical plant consists of three DDA 501-K34 generators, and a continuous internal upper deck exists. All hulls have an SQS-53 sonar dome. Sustained speed - defined as the speed at 80% power - threshold is set to 29.96 knots, the speed DDG51 attains according to the match run. This speed is in fact a constraint, and is further discussed latter in Section 2.12.1.

2.2 General Structure

Computer models for ship design are not new, and appeared immediately after computers became available for the common engineer. Reed [14] wrote a Fortran code in 1976, for his Masters thesis at MIT, with a general structure similar to the DDGx model. The general structure is described in Figure 2.1.

[4] Hull, Mechanical and Electrical

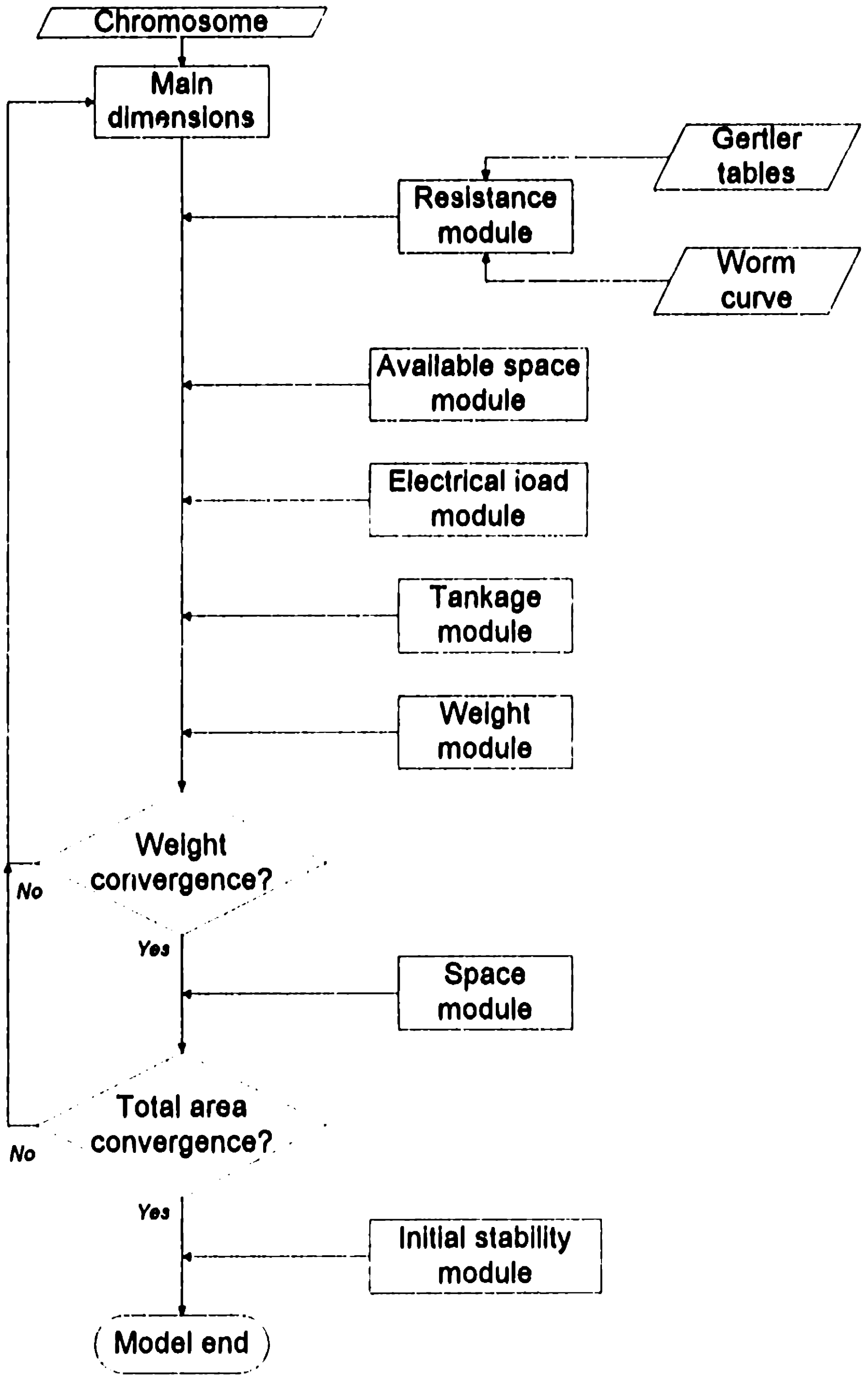


Figure 2.1. *DDGx self-balanced model flowchart.*

The calculations are performed in separate modules, each uses data from its preceding modules, and creates data for the utilization of its succeeding modules. The modules are drawn in Figure 2.1 off the main stream of the process, since they are written as subroutines in the Fortran 90 model. In the Mathcad model all modules, except for one, are located in the main program. The Mathcad model, with the parameters of the DDG51, is shown in Appendix D. The fortran self-balanced model is shown in Appendix F. As can be seen in the flowchart, the computation is performed by the trial and error method. After the desired ship parameters are substituted, the calculation repeats itself iteratively in two loops, until weight and total area convergence is achieved. In Figure 2.1 the genetic notation is used, as the parameters of the ship are represented by a chromosome. Each parameter is considered as a gene.

Two modules are outside the iterative weight loop - space and initial stability modules. The latter is outside both loops. As described in sections 2.10 and 2.11, the execution of these is not required in the displacement balance procedure, and in order to save in running time, is executed only after weight convergence. Area balance is achieved by iterative adjustment of the deckhouse volume. Reed [14] used a raised deck to balance the area requirement.

2.3 Ship Characteristics

Ships evaluated in the DDGx model are uniquely defined by six independent parameters - the genes - together creating the chromosome, as shown in Figure 2.2. The first four are adopted from the math model, while the remaining two are new. The chromosome uniquely defines a specific ship.

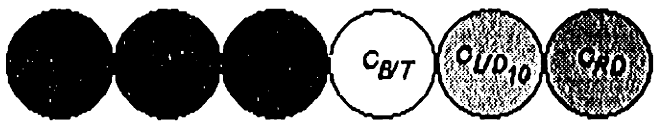


Figure 2.2. *The chromosome.*

Naval architecture parameters are defined in [15] and [16], and are not discussed here. C_P stands for the prismatic coefficient, C_X for maximum section coefficient. $C_{\Delta L}$ represents the displacement to length dimensional ratio, in [lton/ft^3],

$$C_{\Delta L} = \frac{W_{FL}}{\left(0.01 L_{WL}\right)^3} \tag{2.2}$$

where W_{FL} is the full load displacement. Gene $C_{B/T}$ describes the beam to draft ratio, $C_{L/D10}$, the *hull depth coefficient*, defines the non-dimensional ratio of the length on waterline to the depth D_{10}, and finally, the *raised deck coefficient* C_{RD} defines the location of the raised deck. The interpretation of the last two genes is best illustrated by Figure

2.3. The first five genes describe the whole hull, and the last one, C_{RD}, is used to determine the length L_{RD},

$$L_{RD} = C_{RD} L_{WL} \tag{2.3}$$

This length, together with the raised deck height H_R, determine the hull volume which is eliminated from the full hull. A raised deck coefficient of zero implies that no raised deck exists. In all runs a constant raised deck height of 9.5 ft, as in DDG51, is used.

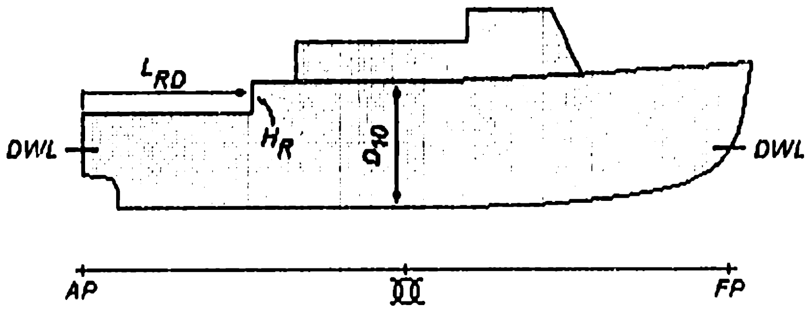


Figure 2.3. *Hull determination.*

Reference [17] describes a method to construct the actual body plan of the vessel as a function of the first five parameters used here. It uses boundary conditions of some existing ship (ASSET has several options) to create the hull lines. In our case DDG51 parent boundary conditions should be employed.

Note that only the hull is defined by the chromosome, and not the superstructure. Since only space-balanced ships are considered feasible, the volume of the deckhouse is a function of the hull, and in fact for a certain chromosome only one unique deckhouse volume is valid. The size of the superstructure is found by the self-balanced model as described in section 2.10. Not all genotypes yield feasible solutions, since there are physical constraints on the shape of the ship and the size of the superstructure. Section 2.12 lists these limitations and deals with them.

2.4 Main Dimensions

At the beginning of the model, see Figure 2.1, an iterative process for the full load weight and total area is established. The computer guesses the displacement W_{FL} and deckhouse volume V_D in the first iteration, or simply uses the values computed in the last one. In each iteration, the main dimensions are calculated according to

$$L_{WL} = 100 \cdot \sqrt[3]{\frac{W_{FL}}{C_{\Delta L}}} \tag{2.4}$$

$$B = \sqrt{\frac{C_{B/T} \nabla_{FL}}{C_P C_X L_{WL}}} \tag{2.5}$$

where ∇_{FL} is the wetted full load volume, and,

$$T = \frac{\nabla_{FL}}{C_P C_X L_{WL} B} \tag{2.6}$$

The depth is computed as in Figure 2.3, by

$$D_{10} = \frac{L_{WL}}{C_{D_{10}}} \tag{2.7}$$

Here $C_{D_{10}}$ is used, instead of $C_{L/D_{10}}$, for abbreviation. Similarly, C_{BT} will be used instead of $C_{B/T}$.

2.5 Resistance Module

The resistance module evaluates the drag of the ship at three speeds: endurance, sustained and maximum. Results obtained by the resistance module are used to check if the sustained speed requirement is satisfied, compute the fuel weight and tank volume. The calculation is conducted according to DDS 051-1 [18], and uses Taylor Standard Series theory [19], with Gertler reanalysis [20]. Useful data was found in the ASSET manual [21], which describes the algorithm ASSET uses. For the sustained and maximum speeds, the procedure is iterative: the computer finds the respective speed by trial and error, until the total drag is balanced by the available power installed aboard the vessel. The algorithm is almost identical to that employed in ASSET, and consequently gives very close results.

2.5.1 Frictional Drag

To calculate the wetted area, [18] provides the chart shown in Figure 2.4, as a function of prismatic coefficient C_P and beam to draft ratio, C_{BT}. [18] neglects dependence on the volumetric coefficient C_V,

$$C_V = \frac{\nabla_{FL}}{L_{WL}^3} \tag{2.8}$$

The graph assumes a TSS[5] hull, but experimentally gives good results for other hulls. Figure 2.4 shows only a portion of the original DDS 051-1 graph, relevant to fast naval ships.

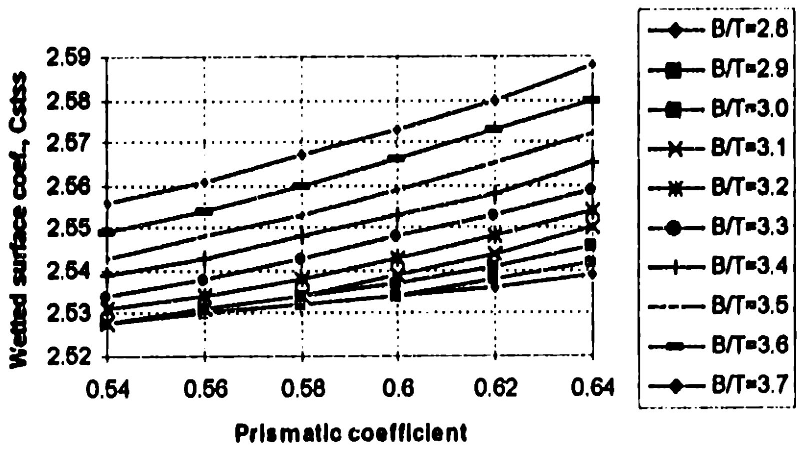


Figure 2.4. *Wetted area coefficient from [18].*

The surface area is calculated by

$$S_{TSS} = C_{STSS}\sqrt{\nabla_{FL} L_{HL}} \tag{2.9}$$

It is possible to sample these curves and build a library file from which the model reads and interpolates the data. However, [21] performed regression analysis, and uses

$$S_{TSS} = C_{STSS}(A_1, A_2, A_3, C_P)\sqrt{\nabla_{FL} L_{HL}} \tag{2.10}$$

Where

$$\begin{aligned} C_{STSS} &= A_0 + A_1 C_P + A_2 C_P^2 \\ A_0 &= 7.028 - 2.331 \cdot C_{BT} + 0.299 \cdot C_{BT}^2 \\ A_1 &= -11 + 5.536 \cdot C_{BT} - 0.704 \cdot C_{BT}^2 \\ A_2 &= 6.913 - 3.419 \cdot C_{BT} + 0.451 \cdot C_{BT}^2 \end{aligned} \tag{2.11}$$

Although [18] uses a non-constant correlation allowance of

[5] Taylor Standard Series

$$C_A = \frac{0.008289}{\sqrt[3]{L_{WL}}} - 0.00064 \tag{2.12}$$

The customary fixed value of 0.0004 is used in the DDGx model, because this number is used in ASSET match run. The frictional drag coefficient is calculated per 1957 ITTC[6],

$$C_F = \frac{0.075}{\left[log(R_N) - 2\right]^2} \tag{2.13}$$

where R_N denotes Reynold's number,

$$R_N = \frac{L_{WL} \cdot V}{\nu} \tag{2.14}$$

The frictional resistance is finally,

$$R_F = \frac{1}{2}\rho\,(S_S + S_{SD})(C_F + C_A)V^2 \tag{2.15}$$

In Equations (2.14) and (2.15) V is the speed, and S_{SD} is the wetted area of the SQS-53 sonar dome. Note that S_S instead of S_{TSS} was used, assuming them to be equal.

2.5.2 Residuary Drag

Unlike the computation of frictional drag, for which the method described in Section 2.5.1 is commonly adopted for all applications, residuary drag can be calculated by various methods, each designated for different hull types. ASSET offers four options for residuary resistance, all explained briefly in [21]. In the DDGx model the reanalysis of Gertler [20] is utilized. First, among the four algorithms in [21], this is the only one which does not require a detailed input, typically not known at this design stage. Secondly, the same procedure is employed in the DDG51 match run of ASSET.

Historically, Gertler conducted his experiment at David Taylor Model Basin at the end of the 1940's and beginning of the 1950's, because the results of Taylor [19], obtained in the same institute approximately five years earlier, are applicable for a narrow range of hulls, for a specific beam to draft ratio. Gertler stayed with TSS hulls, but experimented with them for three different beam to draft ratios. He constructed a large set of graphs for the calculation of the residuary drag coefficient, being a function of four parameters: beam to draft ratio, prismatic coefficient, volumetric coefficient and the dimensional *speed to length ratio*,

[6] International Towing Tank Conference

$$R = \frac{V}{\sqrt{L_{WL}}} \; [knots / \sqrt{ft}] \tag{2.16}$$

Each chart gives residuary drag coefficients as a function of speed to length ratio, for a specific prismatic coefficient, volumetric coefficient and beam to draft ratio. Sampled numerical data, extracted manually from the original curves, was taken from [14]. Unfortunately, for high volumetric coefficients, Gertler checked only a narrow speed to length range, which turned out to be insufficient for fast Naval ships: for C_V of 0.003, 0.004 and 0.005, maximum available speed to length ratio is 1.3 knot/ft$^{0.5}$; for C_V of 0.006, it is even lower, 1.0 knot/ft$^{0.5}$. Only data for volumetric coefficients of 0.001 and 0.002 is complete. The space domain for those four parameters is listed in Table 2.1.

Parameter	Minimal value	Maximal value	In steps of
B/T	2.25	3.75	0.75
C_P	0.48	0.70	0.02
C_V	0.001	0.006	0.001
R $[knt/ft^{0.5}]$	0.5	2.0/1.3/1.0	-

Table 2.1. *Gertler original reanalysis space domain.*

Three extrapolation rules are considered here to extend the Gertler results:

- Linear extrapolation, in which data is extrapolated from the closest lower volumetric coefficient curves, for the same prismatic coefficient and beam to draft ratio.
- Constant spacing extrapolation, in which new data is created for the same C_P and B/T by assuming that at each speed to length ratio the value jumps in constant quantum,

$$C_R(C_V^{i+1}, R) = C_R(C_V^i, R) + [C_R(C_V^i, R) - C_R(C_V^{i-1}, R)] \tag{2.17}$$

- ASSET extrapolation [21]. The drag coefficient corresponding to the highest speed to length ratio available from Gertler for the required volumetric coefficient C_V^j is $C_R(C_V^j, R_{\max}^j)$, where $R_{\max}^j < R$. The drag coefficient for the same speed to length ratio $R_{\max}^j$ but for the closest complete volumetric coefficient C_V^i curve drag coefficient is denoted by $C_R(C_V^i, R_{\max}^j)$, and for the required speed to length ratio R by $C_R(C_V^i, R)$. The extrapolated drag coefficient is:

$$C_R(C_V^j, R) = \frac{C_R(C_V^j, R_{\max}^j)}{C_R(C_V^i, R_{\max}^j)} \cdot C_R(C_V^i, R) \tag{2.18}$$

These methods are illustrated in Figure 2.5 for a selected case of beam to draft ratio of 2.25 and volumetric coefficient of 0.003 (for which data is available only up to speed to length ratio of 1.3 knot/ft$^{0.5}$).

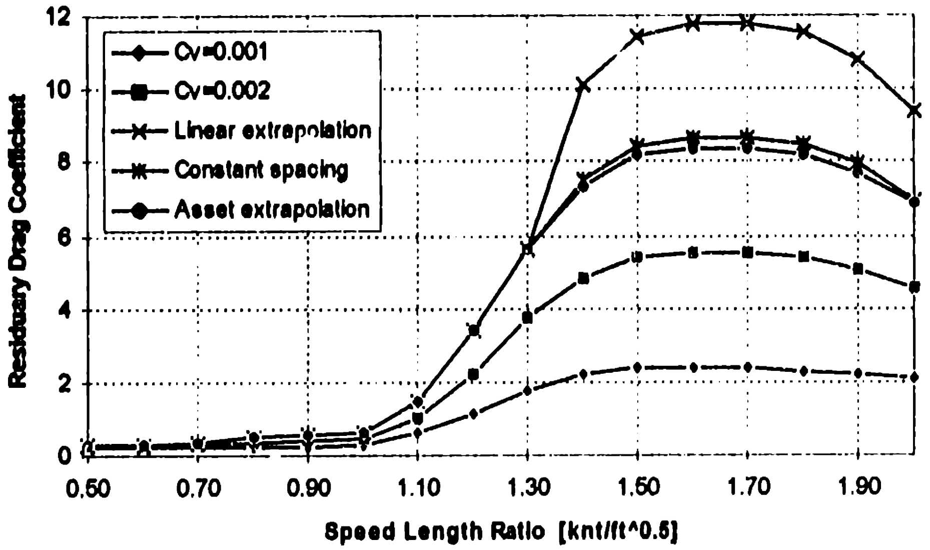


Figure 2.5. *Extrapolation rules for Gertler reanalysis data.*

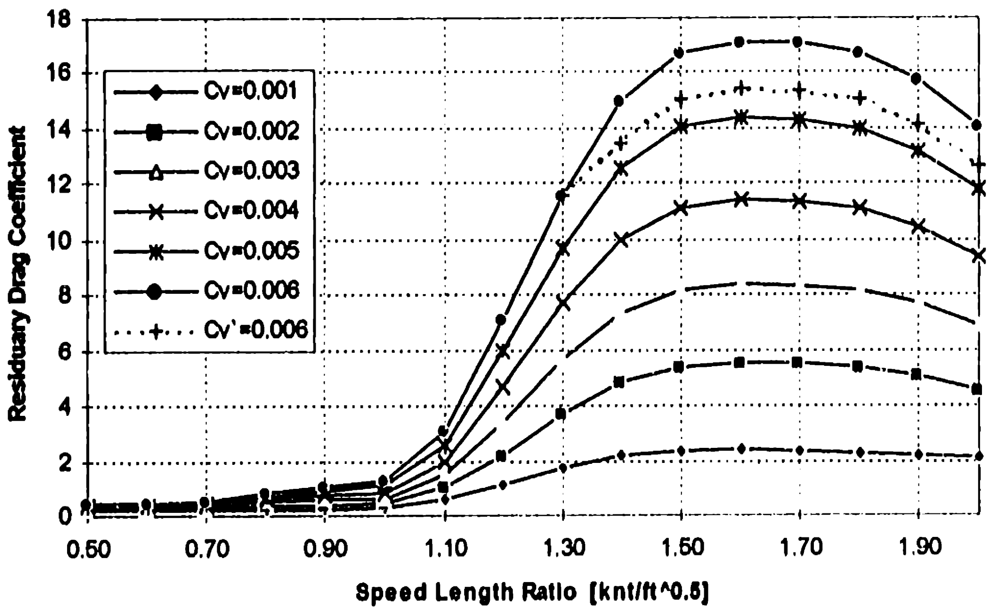


Figure 2.6. *ASSET Extrapolation rule for Gertler's reanalysis, B/T=2.25, C_P=0.48.*

It can be seen that the linear law forms a non-smooth curve, with non-continuous connection. It also seems to give exaggerated drag coefficients. The remaining two methods, on the other hand, look smooth and natural. Because they give similar results, and in an effort to duplicate ASSET results, the latter was chosen for the DDGx model. Figure 2.6 describes the complete extrapolated results, using the ASSET approach, for a specific B/T and C_P. There are two curves for volumetric coefficient of 0.006: a solid and dashed line (denoted $C_V{}'$=0.006). The extrapolation for this parameter departs from [21], since the curve of this method looks inappropriate. In the DDGx model, values for C_V=0.006 are extrapolated twice, using the already extrapolated curve corresponding to C_V=0.005. These are drawn in the solid line. The original curve is dashed. In this way the curve looks more physical and consistent. Complete Gertler residuary drag coefficient extrapolated tables appear in Appendix E. The DDGx model reads the coefficients from these tables.

In order to accommodate hulls with different beam to draft ratio (and not those discrete values tested by Gertler), the *form factor*, given by

$$FF = \frac{4}{3}(C_{BT} - 3) \tag{2.19}$$

is introduced. Values for the drag coefficient are read for all three beam to draft values, with linear interpolation (or end extrapolation if required), and then interpolated again by

$$C_{RTSS} = C_R^{3.00} + FF \cdot \frac{C_R^{3.75} - C_R^{2.75}}{2} + FF^2 \cdot \left(\frac{C_R^{2.75} + C_R^{3.75}}{2} - C_R^{3.00} \right) \tag{2.20}$$

This is an additional departure from ASSET extrapolation, which conducts the procedure by four different sub extrapolations, and then combines the results. The simpler approach of the math model is adopted here. The residual resistance is computed by

$$R_{RTSS} = \frac{1}{2} \rho \, (S_S + S_{SD}) C_{RTSS} V^2 \tag{2.21}$$

However, the DDGx hull is not strictly a TSS daughter hull. In order to account for this deviation, the DDGx model uses the concept of Worm curve. By this approach, the residuary drag is modified by

$$R_R = R_{RTSS} WCF(R) \tag{2.22}$$

Where WCF is the *worm curve factor*. This idea is investigated in [22], in an effort to find an ultimate definition or rule when approaching the problem. The authors conclude that the best and safest way is to perform resistance experiments. Nevertheless, assuming hull similitude with DDG51, worm factors in the DDGx model are taken from the ASSET match run, and the assumption is that the error for different hull parameters is small.

Additional suggestions for values are listed in [21]. The model considers the sonar dome as a part of the hull, not an appendage; the data in ASSET corresponds to that assumption. Figure 2.7 plots worm factors for DDG51.

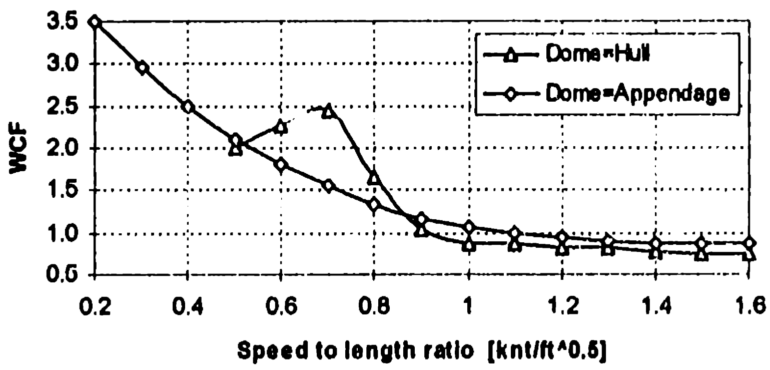


Figure 2.7. *Worm factors for DDG51.*

For comparison, the default Worm factor of [21], for which only the hull (without any dome) is considered, is also plotted. Technically, the values are stored in a library file, and are read by the program. Linear interpolation is performed whenever needed.

2.5.3 Effective Horsepower

The bare hull resistance R_T is the sum of the frictional and residuary resistance, Equations (2.15) and (2.22). The respective power is therefore,

$$P_{EHH} = R_T V \tag{2.23}$$

To this quantity appendages drag and air drag are added.

2.5.3.1 Appendage Effective Power. The algorithm for appendage drag adopted in the DDGx model is quite primitive. ASSET, see [21] and [23], uses a whole set of empirical expressions for each main appendage, such as propellers, rudders, and more. Reference [18] provides Figure 2.8 to estimate the additional drag induced by the appendages. The chart is based on data of existing U.S.N. ships. [18] gives separate charts for single and twin screw vessels, but only the second is shown here. As can be seen, there is a clear distinction between fixed and controllable pitch propellers. This coefficient accounts for all regular appendages in the underwater hull, the propellers in particular. Note that the coefficient has the units of [10^{-5} hp/ft^{3}knot^3]. [18] states that it does not take into consideration special or big items, such as roll fins or sea chests. These need a separate calculation.

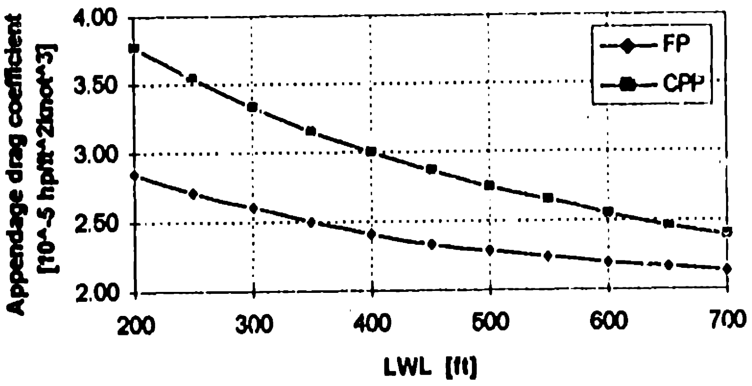


Figure 2.8. *Appendages drag coefficient for twin screw ship, from [18].*

A simple curve fit for a CPP equipped vessel results in,

$$C_{DAPP}[10^{-5}\frac{hp}{ft^2 knot^3}] = -4\cdot 10^{-9} L_{WL}^3[ft] + 9\cdot 10^{-6} L_{WL}^2[ft] - 0.0081 L_{WL}[ft] + 5.0717 \quad (2.24)$$

[18] and [21] introduce the propeller coefficient C_{PROP}, which has the value of 1 for twin screws and 1.2 for a single shaft. The propeller diameter is empirically estimated by [18], [21] and [23] together,

$$D_P = (0.64T + 0.013L_{WL})\cdot C_{PROP} \quad (2.25)$$

Equation (2.25) does not represent the actual diameter of DDG51, but is considered to be an acceptable estimation for early design stages. In the case of roll fins [18] recommends to add 2.5% of the bare hull effective power for each pair. Total appendage power is given by

$$P_{EAPP} = L_{WL} D_P C_{DAPP} V^3 + P_{Efins} \quad (2.26)$$

The original expression in [18] includes a term for a sonar dome, but in this model the dome is considered as a part of the hull, and its contribution is computed in the residuary resistance calculation (Section 2.5.2). During the calibration of the model, to adjust the predictions to those of ASSET, large discrepancies were observed in the output of Equation (2.26). The simplicity of the model relative to the ASSET procedure makes this understandable. A correction factor of 1.23 is incorporated, equating the appendage effective power at 30 knots (sustained speed of DDG51) of the DDGx model with the prediction of the match run. Thus,

$$P_{EAPP} = 1.23 L_{WL} D_P C_{DAPP} V^3 + P_{Efins} \quad (2.27)$$

The calibration relative to the sustained speed condition improves the sustained speed prediction of DDGx model, but has the disadvantage of incorrect resistance values in the endurance mode, which are used to compute the required burnable fuel weight. Because there is a limit on the sustained speed (refer also to Section 2.12.1), the results are calibrated at this speed.

2.5.3.2 Air Effective Power. For the estimation of the frontal area above water line, Figure 2.9 is used.

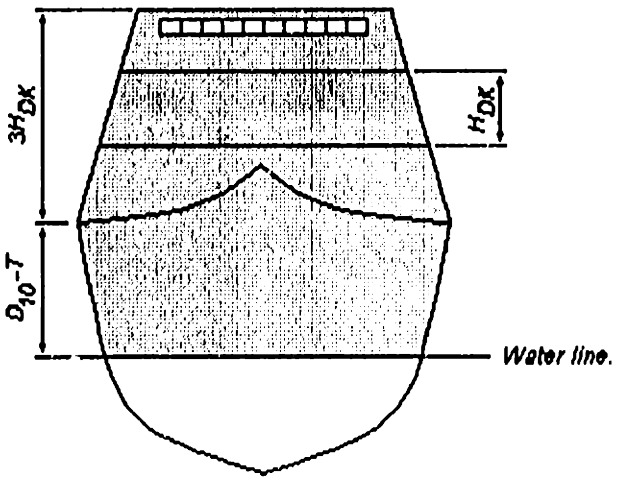


Figure 2.9. *Frontal area calculation.*

By simple geometry, and additional 5% for masts and external equipment,

$$A_{W} = 1.05(D_{10} - T + 3H_{DK}) \cdot B \tag{2.28}$$

The effective air power is

$$P_{EAA} = \frac{1}{2}\rho_A C_{AA} A_W V^3 \tag{2.29}$$

where, as suggested in both [18] and [21], C_{AA}=0.7.

2.5.3.3 Total Effective Power. Using Equations (2.23), (2.27) and (2.29), the overall effective power of the ship is

$$P_{ET} = P_{EBH} + P_{EAPP} + P_{EAA} \tag{2.30}$$

Finally, for design purposes, a *power margin factor* is taken, for which

$$EHP = PMF \cdot P_{ET} \tag{2.31}$$

The DDGx model utilizes a power margin of 8%, as in the match run.

2.5.4 Power Balance

The DDGx model assumes a constant propulsive efficiency of 0.67. Although this coefficient varies in the ASSET match run between this value and 0.70, where it increases as the speed decreases, it gives good results for the relatively low endurance speed. [25] suggests a set of empirical expressions for the propulsive efficiency, as functions of the beam, the draft and the speed, but this is not adopted here. The shaft power is, by definition, where *PC* stands for the propulsive efficiency,

$$SHP = \frac{EHP}{PC} \tag{2.32}$$

A typical mechanical efficiency of 0.97 is chosen. In the sustained speed condition the resistance is increased by 25% (equivalent to 20% reduction in power) for fouling and rough sea state. The required installed power is

$$P_{IREQ} = \frac{1.25 SHP_S}{\eta} \tag{2.33}$$

For the maximum speed, the coefficient 1.25 in (2.33) is omitted,

$$P_{IREQ} = \frac{SHP_{Max}}{\eta} \tag{2.34}$$

These power requirements must be balanced by the propulsion engines, which supply P_{BMAV} horsepower. Recall that the computer balances between P_{BMAV} and (2.33) or (2.34) in an iterative process, with the speed *V* as the variable. In Mathcad™ this process is executed manually, but the Fortran model performs it automatically, using an intrinsic root finder based on the bisection method, see [26] and [27]. A relative error lower than 1% is required for convergence.

2.6 Available Space Module

The available space module assesses the total existing volume in the ship. The output of this module is used to estimate the electrical load (which is a function of the volume of the compartments), the total weight and center of gravity of the vessel, and to check the space balance, versus the space requirements. It uses the results of regression analysis and engineering tools.

The underwater volume is the full load volume, derived from the full load displacement. The vessel's sheer line is a continuous curve passing through three points on the main deck: The two perpendiculars (stations 0 and 20) and midship (station 10). The depth at midship is one of the principle parameters of the hull, and is derived by Equation (2.7). The remaining depths are calculated per DDS 079-2 [28], the Navy standard for freeboard requirements,

$$\begin{aligned} D_0 &= 2.011827T - 6.36215 \cdot 10^{-6} L_{WL}^2 + 2.780649 \cdot 10^{-2} L_{WL} \\ D_{20} &= 0.014 L_{WL}(2.125 + 1.25 \cdot 10^{-3} L_{WL}) + T \end{aligned} \qquad (2.35)$$

The respective freeboard at each station, assuming zero trim, is obtained by subtracting the draft from the depths. The projected side area of the hull is given by

$$A_{PRO} = \frac{L_{WL}}{0.98} \cdot \frac{F_0 + 4F_{10} + F_{20}}{6} \qquad (2.36)$$

where the trapezoidal rule is employed. A correction factor of 0.98 is included, to reflect the average side length (generally the waterline length is 95% of the overall length). The average freeboard and depth are, respectively,

$$\begin{aligned} F_{AV} &= \frac{A_{PRO}}{L_{WL}} \\ D_{AV} &= F_{AV} + T \end{aligned} \qquad (2.37)$$

The average depth is used to calculate the *cubic number* CN. This parameter commonly serves as a statistical measurement of the hull, and is used many times in regression analysis as a parameter. The DDGx model utilizes the cubic number for weight and volume estimation. By definition,

$$CN = \frac{L_{WL} B D_{AV}}{10^5 ft^3} \qquad (2.38)$$

Waterplane coefficient is taken from [15],

$$C_W = 0.278 + 0.836 C_P \qquad (2.39)$$

The above water hull volume is computed according to the illustration in Figure 2.10.

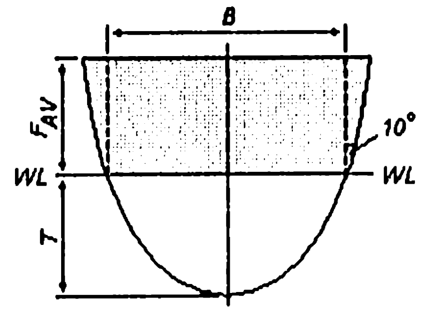


Figure 2.10. *Above water hull volume calculation.*

As in the DDG51, all explored hulls have a 10° flare. Hence, by basic trigonometry,

$$V_{HAW} = F_{AV}^2 tan(10^\circ) L_{WL} + L_{WL} B C_W F_{AV} \tag{2.40}$$

As shown in Figure 2.3, a certain volume should be eliminated from this volume, representing the presence of a raised deck. The breadth of the hull at main deck and first platform are then calculated; the first, B_{MAX}, is the original maximal breadth, and the second, B_{LOW}, is be the breadth at the raised deck area. Using Figure 2.10,

$$\begin{aligned} B_{MAX} &= 2F_{AV} tan(10^\circ) + B \\ B_{LOW} &= B_{MAX} - 2H_R tan(10^\circ) \end{aligned} \tag{2.41}$$

Therefore, the *hull lost volume* due to raised deck is

$$V_{HL} = C_{RD} L_{WL} \cdot \frac{B_{MAX} + B_{LOW}}{2} \cdot H_R C_W \tag{2.42}$$

In (2.42) an average breadth is used. Recall that H_R in (2.41) and (2.42) expresses the raised deck height, and is a predefined magnitude in the DDGx model. Total available hull volume is

$$V_{HT} = V_{HUW} + V_{HAW} - V_{HL} \tag{2.43}$$

Finally, overall ship volume is the sum of the volumes of the hull and the deckhouse,

$$V_T = V_{HT} + V_D \tag{2.44}$$

The total hull volume given in Equation (2.43) is the gross volume; not all of it is available for the requirements of the owner. It includes volume dedicated to machinery and tanks.

2.7 Electrical Load Module

The electrical load module calculates the electrical consumption of all equipment aboard the ship. The module uses the results of the available space module, Section 2.6. Its results are utilized for the calculation of fuel weight, and for the size of the auxiliary machinery rooms. All expressions used are empirical, typically functions of hull volume or crew size. A lot of data was gathered from [25], but the available output ASSET gives is very detailed, and there is no correlation to the expressions listed in [25]. When DDG51 numbers are substituted, the results are totally different. The DDGx model uses empirical expressions from the old 13.412 math model, calibrated to reflect the final results of the ASSET match run of DDG51.

When an electrical balance is performed, the design team checks the load for several scenarios and cruise modes. Traditional cases are battle, cruise, anchor, towing and emergency conditions. Usually battle and cruise conditions are considered in summer and winter. ASSET, as an example, calculates six cases in its machinery module. The 13.412 model computes only the winter cruise condition. The size of the electric plant is determined to satisfy the highest consumption among them. In the majority of cases, either winter battle or winter cruise condition drives the size of the electric plant. For the DDG51 match run the former is higher by less than 5%. Due to the proximity of the results, and in an attempt to simplify the DDGx model, it was decided to use the approach of the 13.412 model. Later it is discovered that the generators are big enough for all hull forms explored, thus this decision is safe.

The empirical expressions for the functional load are not repeated in this section, and can be found in convenient format in the Mathcad model, Appendix D. Most of the categories are calculated directly, but three of them, namely heating, ventilation and air conditioning electrical loads, are functions of the net volume of the ship. As suggested by [25], without the exact numeric values for the constants,

$$\begin{aligned} KW_H &= k_H(V_T - V_{MB} - V_{AUX}) \\ KW_V &= k_V(KW_H + KW_{PAY}) + KW_{CPS} \\ KW_{AC} &= 0.67 \cdot [k_{AC}^1 N_T + k_{AC}^2 (V_T - V_{MB} - V_{AUX}) + 0.1 KW_{PAY}] \end{aligned} \tag{2.45}$$

V_{MB} and V_{AUX} are the volumes of the machinery box and auxiliary machinery rooms, respectively. KW_{PAY} denotes the electrical load of the military payload, KW_{CPS} stands for the consumption of the CPS[7]. All the k's are dimensional constants, used to calibrate the results. For machinery space volume prediction, ASSET uses a large library of propulsion engines and generators, including their physical size and configuration. Volume estimation requires a general arrangement from the user. To simplify the design model, remembering that DDGx and DDG51 are driven by the same HM&E system, the following assumptions are made:

[7] Collective Protection System

- DDGx machinery box volume is the same as at DDG51, for *any* hull chromosome. Since an identical propulsion system is installed, and since the machinery box is the most common location for it, this is a good assumption.
- DDGx volume of auxiliary machinery rooms, which typically consist of the generators and secondary equipment, is linearly related to that of DDG51, according to:

$$V_{AUX}(DDGx) = V_{AUX}(DDG51) \cdot \frac{KW_{MFL}(DDGx)}{KW_{MFL}(DDG51)} \tag{2.46}$$

Thus, an iterative process is established. The loads of Equation (2.45) and the AMR volume are calculated together, in a loop, until convergence is reached. Both Mathcad and Fortran models perform this procedure. The maximum functional electrical load is the sum of all the loads.

The design electrical load, with design and growth margins, is given by

$$KW_{MFLM} = EDMF \cdot EFMF \cdot KW_{MFL} \tag{2.47}$$

As with the power margin, the DDGx model uses values from DDG51: zero design margin, and 1% growth margin. Similarly, the average 24 hour load includes a load margin factor of 20%, and is given in [25]:

$$KW_{24AV} = E24MF \cdot KW_{24} \tag{2.48}$$

Where,

$$KW_{24} = 0.5 \cdot (KW_{MFL} - KW_P - KW_S) + KW_P + KW_S \tag{2.49}$$

KW_P and KW_S are the electrical loads of propulsion and steering systems. These operate non-stop, and are therefore taken out of the influence of the averaging factor. Equation (2.49) is different than suggested in [25] in the averaging factor. [25] recommends on additional averaging factor of 0.8, operating on these two last loads.

2.8 Tankage Module

This module calculates the volume required for all tanks. It reads data evaluated in all prior modules, and its results are used to calculate the net hull volume (the *arrangable* space), the total displacement and center of gravity location. For the fuel computation, the algorithm adopted is from the Navy standard DDS-200-1 [29]. Other liquids are treated empirically, where [25] is the reference. The model only allocates the calculated volume for tankage; it does not arrange the tanks.

2.8.1 Fuel Tanks

Fuel is consumed by the propulsion engines and the generators. The rate depends on the operation of these consumers, and for each machine there exists an economical work point, for which the rate is minimal. The algorithm is taken from [29], without any changes. The fuel carried must enable cruising at the required endurance speed of 20 knots to a distance of 3807.6 nm, the design endurance range. The calculation executed by the model is similar to ASSET. Since the DDG51 and DDGx use a compensated fuel system, the calculation is performed assuming full load displacement during the cruise. The process is divided into two identical sub calculations, each handles the consumption of either the engines or the generators. The total fuel amount is the sum of the results.

2.8.1.1 Propulsion Fuel. This section deals with the fuel required for the propulsion engines. Using the resistance module, the average endurance (meaning at endurance speed) brake horsepower required is, with a 10% margin for fouling and sea state,

$$P_{eBAVG} = 1.1 \cdot \frac{SHP_e}{\eta} \qquad (2.50)$$

The index '*e*' denotes endurance condition. Specific fuel consumption, provided by the manufacturer of the gas turbine, is taken from [25]. The fuel rate at any operational condition is given as a function of the consumption at the maximum continuous rate (that is, at the nominal power), the delivered power and the engine speed (rpm) at that condition,

$$sfc_X = \frac{sfc_{MCR}}{r_p}\left(7.215 \cdot 10^{-2} e^{1.252 r_n} + 0.3629 r_Q e^{0.7229 r_n}\right) \qquad (2.51)$$

The index '*X*' implies that the calculation is for a certain operational condition. sfc_{MCR} is the specific fuel rate at the MCR[8] operational condition, a constant given by the manufacturer. For the LM2500, it has the value of 0.4097 lb/hp·hr. r_p expresses the proportion of the power at the work point to that at MCR,

$$r_p = \frac{P_X}{P_{MCR}} \qquad (2.52)$$

Formerly used P_{BPENG} has been replaced here by P_{MCR}. The next parameter, r_n, describes a similar ratio for rpm. However, the DDGx model does not evaluate this data; ASSET, the source of Equation (2.51), gets it from its Propeller module, based on the predefined series of the propeller. In the absence of this kind of data in the model, the propeller law [24] which states that rpm and speed are proportional is used. ASSET [23] recommends on the same relation,

[8] Maximum Continuous Rate

$$r_{nPE} = \frac{n_X}{n_{MCR}} \approx \frac{V_X}{V_{MAX}} \tag{2.53}$$

Reference [23] is the manual for the initialization module of ASSET. This module was part of previous versions of that package, intended for quick analyses, and was canceled. However, the algorithms listed there are still valid. The last parameter in (2.51) is

$$r_Q = \frac{\frac{P_X}{n_X}}{\frac{P_{MCR}}{n_{MCR}}} \tag{2.54}$$

Substitution of (2.52) and (2.53) into (2.54) reveals that

$$r_Q = \frac{r_p}{r_n} \tag{2.55}$$

Since endurance fuel is required, Equations (2.51) through (2.55) are used for the endurance cruise condition, and the specific fuel rate of the propulsion engines at endurance speed, sfc_{ePE}, is obtained. [29] instructs to correct this value for instrumentation inaccuracy (during ship acceptance trials) and machinery design changes (accounts for minor machinery changes during construction), by the correction factor

$$f_1 = \begin{cases} 1.04 & if \quad 1.1SHP_e \le \frac{1}{3}\cdot\frac{P_l}{2} \\ 1.03 & if \quad 1.1SHP_e \ge \frac{2}{3}\cdot\frac{P_l}{2} \\ 1.02 & \quad Otherwise \end{cases} \tag{2.56}$$

$P_l/2$ expresses the nominal power of the two operating gas turbines at the endurance condition. The option to operate only two gas turbines at endurance speed, with one trailed propeller, is neither considered here nor in the DDG51 match run. As can be seen, the correction factor depends on the proximity of the shaft horsepower to the brake horsepower. [25] has modified Equation (2.56), converting it to a continuous function, by passing a straight line between the values of 1.04 and 1.03; the DDGx model adopts the original version of the standard. The specific fuel rate is also margined by an additional safety factor of 5%, for plant deterioration,

$$FR_{AVG} = 1.05\cdot f_1 \cdot sfc_{ePE} \tag{2.57}$$

This is the average fuel rate, in [lb/hp·hr]. The *burnable* propulsion fuel weight is then

$$W_{BP} = \frac{E}{V_e} P_{eBAVG} FR_{AVG} \tag{2.58}$$

2.8.1.2 Electrical Fuel. Here the amount of fuel required for the generators is evaluated. The algorithm is very similar to that of Section 2.8.1.1. The main difference is the conversion from electrical load to mechanical power. Again, [25] with its machinery library is used. According to factory data for the DDA 501-K34 generator gas turbine, a desired electrical power KW_{24AVG} is converted to mechanical power by the relation

$$P_{G24} = KW_{24AVG} / a \tag{2.59}$$

where a is a conversion factor that accounts for physical units and efficiencies. A value of 0.7087 kW/hp is given in [25]. The process described in the last section is now revisited. According to the ASSET DDG51 match run, each generator turbine generates 4600 hp at MCR, so (2.52) becomes

$$r_p = \frac{P_{G24} / N_G}{4600} \tag{2.60}$$

The division of the mechanical power by N_G (=2) is because two generators are used for all applications. The third generator is for emergency. For lower fuel consumption, the engine speed is set to the rated rpm, consistent with a constant generator frequency. Hence r_{nG}=1 for any work point. The specific fuel rate at MCR is 0.4727 lb/hp·hr. The formula for the specific fuel rate at a specific work point for this turbine is a polynomial rather than exponential,

$$sfc_X = \frac{sfc_{MCR}}{r_p}(0.2821 + 0.7179 r_Q) \tag{2.61}$$

When the results from the electrical load module are plugged into Equations (2.59) through (2.61), sfc_{G24}, the specific average fuel rate per hp, in [lb/hp· hr], is obtained. The same flow rate, per kW, is

$$sfc_{GE24} = \frac{sfc_{G24} P_{G24}}{KW_{24AVG}} \tag{2.62}$$

It has the units of [lb/kW· hr]. The same margins of [29] are applied, only that here f_i has a constant value of 4%. Thus,

$$FR_{GAVG} = 1.04 \cdot 1.05 \cdot sfc_{GE24} \tag{2.63}$$

Similar to (2.58) the burnable electrical fuel weight is calculated by

$$W_{Be} = \frac{E}{V_e} K W_{24AVG} FR_{OAVG} \tag{2.64}$$

2.8.1.3 Total Fuel Weight. The actual required fuel weight is divided by the tailpipe allowance *TPA*, with the value of 0.95 commonly used,

$$W_{FP} = \frac{W_{BP}}{0.95} \qquad W_{Fe} = \frac{W_{Be}}{0.95} \tag{2.65}$$

Total required fuel weight is the sum of the weights of (2.65),

$$W_{F41} = W_{FP} + W_{Fe} \tag{2.66}$$

F41 is the SWBS group for fuel. See next section for a description of the SWBS classification system. The volume of the fuel tanks, with a 2% addition for internal structure and 5% for expansion, is

$$V_F = 1.02 \cdot 1.05 \cdot W_{F41} \cdot \gamma_F \tag{2.67}$$

γ_F denotes the specific volume of fuel, 42.3 $ft^3/lton$.

2.8.2 Other Tanks

This section describes the algorithms for tanks other than fuel. Here some of the expressions are logical, based on engineering, and others are empirical. Almost all of them are taken from [30].

2.8.2.1 Helicopter Fuel. The weight of this fuel, 64.4 lton (see Appendix C), is part of the military payload the ship has to carry. Using the methodology of Equation (2.67), the required volume is given by

$$V_{HF} = 1.02 \cdot 1.05 \cdot W_{F42} \cdot \gamma_{HF} \tag{2.68}$$

F42 is the SWBS group of helicopter fuel. γ_{HF} is 43 $ft^3/lton$.

2.8.2.2 Lubrication Oil. ASSET, as described in [25], has empirical expressions for the oil amounts required by different machinery. The DDGx model uses the oil weight of DDG51, 17.6 lton, since the same HM&E system as DDG51 is installed. This assumes the same amount for any hull shape. Hence,

$$V_{LO} = 1.02 \cdot 1.05 \cdot W_{F46} \cdot \gamma_{LO} \tag{2.69}$$

where γ_{LO}=39 $ft^3/lton$ is used.

2.8.2.3 Potable Water. The requirement is to carry 0.15 lton of fresh water for each crew member. Therefore, the water weight is, where N_T represents the crew size,

$$W_{F52} = N_T \cdot 0.15 \quad [lton] \tag{2.70}$$

Water expansion is negligible, so the required volume is

$$V_W = 1.02 \cdot W_{F52} \cdot \gamma_W \tag{2.71}$$

ASSET results for fresh water weight in the match runs are a little lower, but in this case the results have not been calibrated to fit the DDG51.

2.8.2.4 Sewage and Waste Oil. From [30],

$$\begin{aligned} V_{SEW} &= (N_T + N_A) \cdot 2.005 \quad [ft^3] \\ V_{WASTE} &= 0.02 \cdot V_F \end{aligned} \tag{2.72}$$

N_T+N_A expresses the total number of accommodations on the vessel. V_F is the fuel tank total volume, taken from Equation (2.67).

2.8.2.5 Clean Ballast. DDGx ships are equipped with a compensated fuel system, as is DDG51. Compensated systems require some ballast for heel and trim adjustments. From the match run and [30],

$$V_{BAL} = 0.19 \cdot V_F \tag{2.73}$$

In ASSET the 0.19 coefficient is called the ballast fuel fraction.

2.8.3 Total Volume of Tanks

Total volume allocated by the DDGx model for tanks is the sum of all volumes, Equations (2.67) through (2.73), or,

$$V_{TK} = V_F + V_{HF} + V_{LO} + V_W + V_{SEW} + V_{WASTE} + V_{BAL} \tag{2.74}$$

2.9 Weight Module

The weight module computes lightweight and full load displacement, and their respective center of gravity locations. As illustrated in Figure 2.1, the module closes the weight loop: It compares the initial full load weight to the final one, until convergence within a relative error of 1% is reached. The module reads necessary data from all its preceding modules, and after convergence, the results are used by the initial stability module. The iterative process is conducted according to the substitution basic method, described in [27]: In case of relative error larger than the assigned limit, the process is repeated with the final value as the guess. This method is not guaranteed to converge, but is very useful due to its simplicity. The Fortran model is limited to 15 iterations. In the rare case of divergence, the model assigns the chromosome a payload fraction value of zero. For convenience, the SWBS[9] weight classification method is adopted. This method is weight-based. It breaks the weights into seven main groups (100-hull, 200-machinery, 300-electric plant, 400-command and surveillance, 500-auxiliaries, 600-outfit, 700-armament) and other groups for variable payloads. ASSET manuals provide empirical expressions for the weight and center of gravity of all 3 digit SWBS items on board, but this is too detailed and complex for the DDGx model. Instead, the math model's original weight module is kept. The weight is divided into the seven main groups, but each group consists of not more than five categories. This classification is primitive, but the calculation is significantly simpler and straight forward. Whenever possible, ASSET expressions are used, replacing the old ones. As a rule, all the expressions are re-calibrated, to give the same results as the output of ASSET weight module for DDG51.

In the DDGx weight distribution, group 100, hull weights, is divided into the basic hull, consisting of groups 110 through 140, 160 and 190, the deckhouse, group 150, the masts, group 171, and foundations, group 180. The deckhouse is made of steel, as required by the U.S. Navy. The DDGx model uses the same masts as the DDG51 for all runs, with a weight of 2 lton. Group 200 is distributed into basic machinery, accounting for groups 239, 241, 242, and 250 through 290, shafting, group 243, propulsors, 245, and bearings, group 244. SWBS group 300 is not divided into any sub-groups. The command and surveillance, group 400, is divided into gyro/internal communication/navigation, groups 420 and 430, and cabling. Most of the weight of this group comes from the military payload. Group 500 is generally not distributed. Group 600 is divided into two sub-groups: hull fittings, groups 610, 620, 630, and personnel related outfit, groups 640 through 670. Group 700 is entirely determined by the payload. The empirical expressions are not listed here, and can be found in Appendix D.

After all weights are computed, the lightweight is calculated. Although the standard weight margin for concept design is 10%, a weight margin factor WMF=0.5% is added, as in the match run of the DDG51. The weight margin for future growth is

$$W_{M24} = WMF \cdot \sum W_i \qquad (2.75)$$

W_{M24} is the only weight margin employed by the model. For the center of gravity calculation, the assumption is that the center of gravity of the margin coincides with the lightship center of gravity. The summation includes the weight of the payload (without

[9] Ship Work Breakdown Structure

the variable payload, which is not part of the lightweight) corresponding to each weight group. The lightweight is

$$W_{LS} = \sum W_i + W_{M24} \tag{2.76}$$

Next, the variable load is computed. The weight of provisions, SWBS group F31, and general stores, group F32, depends on the crew size and the required stores period. The general form of the equations of [32] is used, with re-scaled constants, as appears in Appendix D. The weight of the crew, group F10, uses the expression

$$W_{F10} = 236 \cdot N_E + 400 \cdot (N_O + 1) \qquad [lb] \tag{2.77}$$

(2.77) assumes that officers are allowed to have a more baggage. The total full load weight is the sum of the lightweight and all other weights,

$$W_T = W_{LS} + W_{VP} + W_{F41} + W_{F46} + W_{F52} + W_{F31} + W_{F32} + W_{F10} \tag{2.78}$$

W_{VP} represents the variable payload, the part of the payload which is changeable. DDGx variable payload consists of ammunition and helicopter fuel. The calculation of the center of gravity is conducted in the regular way, where empirical locations (except for the fuel, which is assumed to have a constant center of gravity location of 10.3 ft above base line) are assigned to each weight sub-group, and the moments about the baseline are summed. A margin of 0.5 ft is added to the center of gravity at full load condition. The expressions here are entirely empirical as well, and are scaled to give the numbers of DDG51.

Convergence of the weight loop is reached when the guessed full load displacement, W_{FL}, and the resulted weight, W_T, coincide within 1%.

2.10 Space Module

The space module determines the space required to be available in the ship. The calculation is performed by summing areas: Appendix C lists area required for the military payload, while other necessary areas are evaluated using empirical expressions. As in the case of weight, the original approach of the math model and not the detailed computation of ASSET is adopted. However, all expressions are modified. The actual expressions are not repeated in this section, and are listed in Appendix D.

Required areas are divided into two types: equipment which must be installed in the deckhouse, and equipment which can be placed in either the hull or superstructure. For example, the bridge has to be located in the deckhouse, but stores can be anywhere in the ship. The military payload in Appendix C uses the same classification. The model output is the deckhouse required area A_{DR}, the hull or deckhouse required area A_{HR}, and the total area required, A_{TR}, where

$$A_{TR} = A_{HR} + A_{DR} \tag{2.79}$$

A vessel which does not have enough area, is not feasible, and considered to be in imbalance. Surprisingly, while DDG51 is a real ship, its match run is not space-balanced; the computer sends a massage that the required area is higher than the available by approximately 10000 ft^3. Because the ship is assumed to be space balanced, and since the results for the available space of the model and ASSET match run are very similar, the required space is calibrated such that the DDG51 run in DDGx model is space-balanced, and not according to ASSET space module results.

As shown in Figure 2.1, the model balances total area by adjusting the deckhouse volume such that the required area equals the available net area. The available space module computes the gross hull volume, Equation (2.43). The gross value consists of machinery and tankage spaces, which are not available for other applications. The net available hull volume, representing the overall allocation-entitled volume, is

$$V_{HA} = V_{HT} - V_{MB} - V_{AUX} - V_{TK} \tag{2.80}$$

where V_{MB} stands for the main machinery rooms volume (typically referred as machinery box), V_{AUX} for the auxiliary machinery rooms volume, Equation (2.46), and V_{TK} for the tanks volume, taken from (2.73). The superstructure is considered to be entirely available for space allocation, although it would be bad engineering judgment to place machinery equipment or tanks up there. Hence, the net total arrangable volume is

$$V_{TA} = V_{HA} + V_D \tag{2.81}$$

The net total available area is therefore, where H_{DK} describes the average deck height,

$$A_{TA} = \frac{V_{TA}}{H_{DK}} \tag{2.82}$$

To minimize weight, the vessel is required to be area-balanced. Otherwise, either the ship is too small to enclose all required facilities, or too big, and therefore heavier than needed. Unlike weight, no margin is preserved for future growth. The area balance equation is constructed with the deckhouse volume as the unique argument,

$$Bal(V_D) = A_{TR} - A_{TA}(V_D) = 0 \tag{2.83}$$

and then solved by the computer using [34], an intrinsic function in Fortran 90. An initial guess of 190000 ft^3 is taken, only because this is the volume in DDG51. The limit is set to 20 iterations. Usually convergence is reached in less than 8 trials, and in the unlikely case of divergence, the payload fraction of that vessel is assigned the value of zero. Convergence is declared when A_{TR} and A_{TA} are within a relative error of 1%.

2.11 Initial Stability Module

The initial intact stability module calculates the ratio of metacentric height and the beam, $C_{GM/B}$. It uses data from the weight and available space modules. This value is traditionally used to assess stability of ships. The module is classified as initial, because it expresses the stability for small heel angles, a feature which depends on the metacentric height. For small angles this is an accepted analysis; actually, the righting arm curve of this basic analysis is tangent to the rigorous analysis curve at small roll angles, as shown in [15] and [16].

The algorithm is taken from [15], and is one of the basic calculations in naval architecture. The vertical center of buoyancy is estimated by the Moorish formula,

$$KB = \frac{T}{3} \cdot \left(2.4 - \frac{C_P C_X}{C_W} \right) \tag{2.84}$$

where C_W is taken from (2.39). The original formula uses 2.5 in the parenthesizes, but is modified to 2.4 to fit the results of DDG51. The distance from the center of buoyancy to the metacenter is, for small angles,

$$BM = \frac{I_{XX}}{\nabla} \tag{2.85}$$

Here I_{XX} is the transverse water plane moment of inertia, and ∇ the underwater displaced volume. For the determination of the moment of inertia, the *transverse water plane inertia coefficient* is introduced,

$$C_{IT} = \frac{I_{XX}}{\frac{L_{WL} B^3}{12}} \tag{2.86}$$

This coefficient represents the ratio between the actual moment of inertia to the moment of inertia of a rectangle with the dimensions of $L_{WL} \cdot B$. This coefficient is empirically given in [15],

$$C_{IT} = -0.537 + 1.44 \cdot C_W \tag{2.87}$$

As before, this expression is modified, to reflect the actual results of DDG51. The metacentric height is

$$GM = KB + BM - KG \tag{2.88}$$

Finally, the desired stability criterion is

$$C_{GM/B} = \frac{GM}{B} \qquad (2.89)$$

2.12 Feasibility Constraints

Not all solutions from the DDGx self-balanced model are physically possible, or meet the design requirements specified in Section 2.1. A non-possible solution is called non-feasible. Mathematically, each suggested solution is expected to meet a set of constraints, some of which are equalities, and others are inequalities. Two equalities exist, the total area constraint and the weight constraint, described earlier in this chapter. It is clear that a satisfaction of an equality constraint is more complex in reference to an inequality constraint, because an equation has a finite number of solutions, while the second has an infinite number of them. For this reason the equality constraints are satisfied automatically by the Fortran model. All other constraints are only checked by the DDGx model after the evaluation of the ship is completed. A feasibility status is printed at the end of the evaluation. No effort is performed by the model in order to make the vessel feasible.

The set of constraints is presented in this section, in the order of appearance of the associated module. Some limitations of the model are derived from a local design decision, and are not applicable for all general cases. For instance, the requirement for a continuous internal deck - the damage control deck - limits the height of the machinery box for a specific depth D_{10}. This limit does not exist for ships with discontinuous first deck.

2.12.1 Sustained Speed Constraint

Section 2.1 sets the minimal accepted sustained speed to 29.96 knot, hence

$$V_s \geq 29.96 \quad knots \qquad (2.90)$$

where V_S is calculated in the resistance module. The exact sustained speed of DDG51 (instead of 30 knots) is used as the constraint, in order to increase the number of feasible ships in the population. It is found later that the overall number of feasible chromosomes, in the solution space defined in Chapter 4, is considerably low.

2.12.2 Depth Constraint

The main deck sheer line is determined by the available space module. The value for the depth, D_{10}, is a function of the fifth gene in Figure 2.2, the hull depth coefficient.

However, not any value is allowed. Reference [33] requires that at 25° heel the deck edge must be out of the water. Using simple trigonometry, as illustrated in Figure 2.11,

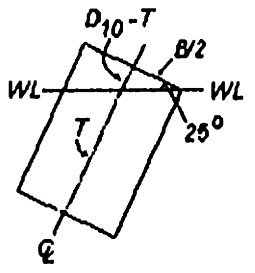


Figure 2.11. *Heel constraint for depth.*

$$\frac{D_{10} - T}{\frac{B}{2}} = tan(25^0) \quad \text{Or,} \quad D_{10} = 0.233B + T$$

However, this calculation assumes that the ship heels about the center of floatation, which is an acceptable hypothesis for small angles only, not 25°. Also, the hull may have a bulwark, so artificially the deck edge is higher. Hence, a somewhat more lenient expression is adopted, from the original math model,

$$D_{10} \geq 0.21B + T \tag{2.91}$$

Additionally, the hull must be structurally strong enough to withstand various sea loads. The DDGx model does not compute stresses and section modulus, so a very simple expression, from the math model as well, is used,

$$D_{10} \geq \frac{L_{WL}}{15} \tag{2.92}$$

This relation was derived from data of existing ships. In the two last equations, (2.91) and (2.92), H_R is added if C_{RD} is greater than 0.5, since in this case the actual (or *effective*) depth is D_{10}-H_R. Finally, the design requires a continuous deck for better damage control, and, assuming that the machinery box is near midship, the depth must also satisfy the inequality

$$D_{10} \geq H_{MB\min} + H_{DK} \tag{2.93}$$

Otherwise, the passage above the main machinery rooms is not accessible. $H_{MB\min}$ represents the minimal machinery box allowed, taken from [25]. This value, 22 *ft* in

DDGx case, is a function of the propulsion engine type. Equation (2.93) ensures that all machinery fit into the machinery box. In Equation (2.93) there is no need to account for H_R, because even if a raised deck exists, its external part can be used as damage control deck, a continuance of the internal damage control deck. Practically, the model assigns the maximal value among Equations (2.91), (2.92) and (2.93) to the parameter D_{10min}, and the constraint becomes

$$D_{10} \geq D_{10\min} \qquad (2.94)$$

2.12.3 Electrical Plant Constraint

DDGx ship electrical plant consists of three DDA 501-K34 generators, each produces 2500 kW. Regularly, only two of them are operated, and the third is used as a standby generator, for emergency. The electrical load module computes the maximum functional electrical load, including margins. This parameter represents the highest electrical consumption of the ship. This amount can be evaluated per generator,

$$KW_{GREQ} = \frac{KW_{MFLM}}{(N_G - 1)\cdot 0.9} \qquad (2.95)$$

where KW_{MFLM} is the maximal possible load, N_G the number of generators installed, and an additional margin to compensate for voltage fluctuations. Equation (2.95) expresses the maximum electrical load required from each operating generator. Hence, the electrical load constraint is

$$KW_G \geq KW_{GREQ} \qquad (2.96)$$

where KW_G is the nominal power of one generator (2500 kW in DDGx and DDG51). Experimentally, this constraint is easily met for all hull parameter combinations, since the installed electric power aboard the vessel is large.

2.12.4 Deckhouse Area Constraint

In Section 2.10 two types of area allocation are introduced: area of equipment which must be allocated in the deckhouse, and area which can be placed in either the deckhouse or the hull. The first is referred as deckhouse area, and the latter hull area, where it should be born in mind that it means hull *or* deckhouse. This means that if the hull is too small, a portion of its required area can be transferred to the superstructure; theoretically, deckhouse volume can be increased infinitely, until area balance is reached. This is not always feasible, obviously, and one tool to restrict it is the deckhouse volume constraint, described later in Section 2.12.5.

The solution for the superstructure volume, Equation (2.83) is merely mathematical. Therefore, a negative deckhouse volume can be the output of the program. The net total available volume was shown to be

$$V_{TA} = V_{HA} + V_D \tag{2.81}$$

Or, dividing by the average deck height,

$$A_{TA} = A_{HA} + A_{DA} \tag{2.97}$$

Thus, a negative deckhouse volume implies that the opposite happened: the net available hull area is higher than the required hull area, in an amount that also covers the requirements of deckhouse area, so, mathematically, there is no need for a superstructure. The total area constraint is thus not enough, and its satisfaction, even if all other limitations are met, does not result in a feasible ship. To correct this, a secondary area constraint is required: the available deckhouse area must be greater than the required deckhouse area,

$$A_{DA} \geq A_{DR} \tag{2.98}$$

In an event of negative deckhouse volume, the available area is negative as well, since

$$A_{DA} = \frac{V_D}{H_{DK}} \tag{2.99}$$

and (2.98) cannot be satisfied. The deckhouse area constraint can be also considered as a minimum deckhouse volume constraint, because Equation (2.98) may be written as

$$\frac{V_D}{H_{DK}} \geq A_{DR}$$

or,

$$V_D \geq H_{DK} A_{DR} = V_{D\min}$$

2.12.5 Deckhouse Volume Constraint

The superstructure volume must also be within reasonable positive limits. The DDGx model evaluates a superstructure with two decks, and an additional one for the bridge only. The typical shape, with maximum allowed dimensions, is illustrated in Figure 2.12. For RCS[10] reduction, a serious consideration in modern war ships, superstructure

[10] Radar Cross Section

sides are inclined by 10°. The maximum width is the beam B, such that the side clearance (to the deck edge) is zero, and the passage is placed in the structure. This is the recommended arrangement for a reduced RCS, also.

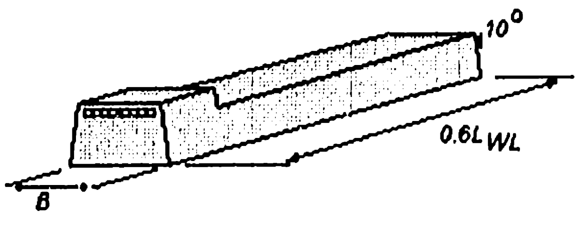


Figure 2.12. *Maximal deckhouse dimensions.*

The length limit is 60% of L_{WL}, leaving space for a helicopter landing area, deck weapons and mooring operations. Adding 10% for the bridge,

$$V_{D\max} = \left(2B - 4\frac{H_{DK}}{tan(80^0)}\right)\cdot H_{DK}\cdot 0.6L_{WL}\cdot 1.1 \tag{2.100}$$

where simple stereometry is used. The deckhouse constraint is, then,

$$V_D \leq V_{D\max} \tag{2.101}$$

2.12.6 Initial Stability Constraints

Stability is considered in the DDGx model in a very basic approach, applicable for small heel angles only. The stability measure is the metacentric height to beam ratio, $C_{GM/B}$. Naturally, a ship must have a sufficient residual righting arm (see [15] and [16]), providing a moment to resist the heel motion. Proceeding with the same small heel angle analysis, it can be shown that heel motion is sinusoidal, with a period of

$$T = \frac{2\pi i}{\sqrt{g\cdot GM}} \tag{2.102}$$

i is the mass longitudinal radius of gyration. (2.102) shows that the period decreases as the metacentric height is increased. However, an exaggerated value of GM makes the period too short; the ship is stable, but too stiff. The hull motion has a low roll period, making it uncomfortable for the crew, and inappropriate for normal function. In such a case the ship is said to be rigid. The parameter $C_{GM/B}$, as a result, is bounded by upper and lower limits.

From the experience of the U.S. Navy, while evaluating stability per [33], the lower limit is a ratio of 0.09. This number is also presented in the MIT math model, and adopted in the DDGx model. Equation (2.102) can also be presented in an empirical relation, taken from [16],

$$T = \frac{cB}{\sqrt{GM}} \tag{2.103}$$

c is a dimensional constant, between 0.72 and 0.91 sec/m$^{0.5}$. Rearranging (2.103),

$$\frac{GM}{B} = \frac{c^2 B}{T^2} \tag{2.104}$$

DDGx has a beam of around 20 meters. When a minimal roll period of 10 seconds is required, and extreme values of the coefficient c are substituted, Equation (2.104) gives extreme values for the rigidity bound of $C_{GM/B}$,

$$0.103 \le (C_{GM/B})_{rigid\ stability} \le 0.165$$

When the recommended value for frigates, 0.8 sec/m$^{0.5}$, is plugged in, the result is an upper limit of 0.128. For the DDGx model an upper value of 0.135 is used for $C_{GM/B}$, vice the MIT model's value of 0.122. Thus, the initial stability constraint is

$$0.09 \le C_{GM/B} \le 0.135 \tag{2.105}$$

2.13 Comparison to DDG51

Figure 2.14 describes a typical self-balanced model output, run with DDG51 hull chromosome.

```
Main Parameters ...

  Cp                                                 .609
  Cx                                                 .819
  Cdl                                               80.68  [lton/ft^3]
  Cbt                                              2.926
  CD10                                             11.13
  Crd                                               .200

  Main Dimensions on Waterline ...

  Underwater volume                                2.846E+05 [ft^3]
  Length on waterline                             465.44  [ft]
  Beam                                             59.89  [ft]
  Draft                                            20.47  [ft]
  D10                                              41.82  [ft]
```

```
Speed and Resistance ...

Wetted area                              30726.5 [ft^2]
Sustained speed                          30.00 [knt]
Maximal speed                            31.64 [knt]
Endurance shaft horsepower               14942.1 [hp]

Space Available ...

D0                                       52.74 [ft]
D10                                      41.82 [ft]
D20                                      38.11 [ft]
Total hull volume                         786783.3 [ft^3]

Electrical Loads ...

Winter cruise electrical load            3443.6 [kW]
Marginal winter cruise electrical load   3478.1 [kW]
24 hours average electrical load         2313.2 [kW]
Power required per generator             1932.3 [kW]
Auxiliary machinery rooms volume         57444.4 [ft^3]

Tankage ...

Fuel weight                              1171.5 [lton]
Fuel tanks total volume                  53071.0 [ft^3]
Ballast tank volume                      10083.5 [ft^3]
Total tanks volume                       70568.0 [ft^3]

Weight and Center of gravity ...

Lightweight                               6428.0 [lton]
Vertical CG of lightship                 25.00 [ft]
Full load displacement                    8091.1 [lton]
Vertical CG at full load                 23.10 [ft]
Payload fraction                          .113

Area/volume Balance ...

Required hull area                       56587.1 [ft^2]
Available hull area                      48794.6 [ft^2]
Required hull volume                     603217.9 [ft^3]
Available hull volume                    520150.9 [ft^3]
Required deckhouse area                  10207.5 [ft^2]
Available deckhouse area                 17542.9 [ft^2]
Required deckhouse volume                108811.5 [ft^3]
Available deckhouse volume               187007.4 [ft^3]
Total required area                      66794.5 [ft^2]
Total available area                     66337.6 [ft^2]
Total required volume                    712029.5 [ft^3]
Total available volume                   707158.3 [ft^3]

Initial Stability .....

KB                                       12.05 [ft]
BM                                       17.46 [ft]
Metacentric height                        6.41 [ft]
GM to B ratio                             .107

Deckhouse volume                         188505.4 [ft^3]

Balance/Feasibility Status ...

Category                             Required/Minimal    Available
Deckhouse area [ft^3]                     10207.5          17542.9
Electric plant [kW]                       1932.3           2500.0
```

```
Sustained speed                          29.96          30.00
Initial stability                         .090          .1071
                                          .135
Depth [ft]                               33.04          41.82
Maximal deckhouse volume [ft^3]          367610.9       188505.4
Payload fraction                          .075          .1134
```

Figure 2.13. *Self-balanced DDGx model output, with DDG51 hull parameters.*

Table 2.2 compares the results of ASSET and the DDGx model for the DDG51.

	Units	ASSET	Model	Error [%]
Hull chromosome				
Prismatic coefficient	-	0.609	0.609	-
Midship coefficient	-	0.819	0.819	-
Displacement to length ratio	[lton/ft³]	80.68	80.68	-
Beam to draft ratio	-	2.93	2.93	-
Length to depth ratio	-	11.13	11.13	-
Raised deck coefficient	-	0.2	0.20	-
Main dimensions				
L	[ft]	465.73	465.44	-0.06
B	[ft]	58.84	59.89	+1.78
T	[ft]	20.11	20.47	+1.79
D0	[ft]	52.06	52.74	+1.31
D10	[ft]	41.83	41.82	-0.02
D20	[ft]	36.29	38.11	+5.02
Length to beam ratio	-	7.92	7.77	-1.81
Water plane coefficient	-	0.787	0.787	0.00
Resistance				
Sustained speed	[knt]	29.96	30.00	+0.13
Maximal speed	[knt]	31.28	31.64	+1.15
Endurance shaft horsepower	[hp]	14731.0	14942.1	+1.43
Wetted surface area	[ft²]	29298.4	29326.5	+0.10
Electrical Loads				
Winter cruise maximal load	[kW]	3439	3478.1	+1.14
Average 24 hour electrical load	[kW]	2363.3	2313.2	-2.12
Space				
Hull volume	[ft³]	800691	786783.3	-1.74
Deckhouse volume	[ft³]	192009.4	188505.4	-1.82
Average hull deck height	[ft]	10.66	10.66	-
Total required area	[ft²]	78368	66794.5	-14.77
Total available area	[ft²]	68658	66337.6	-3.38
Weight				
Fuel weight	[ton]	1186.7	1171.5	-1.28
Light ship weight	[lton]	6453.1	6428.0	-0.39
Light ship VCG	[ft]	25.0	25.0	0.00
Full load weight	[lton]	8126.9	8091.1	-0.44
Full load VCG	[ft]	22.69	23.10	+1.81
Full load KB	[ft]	11.94	12.05	+0.92
Full load BM	[ft]	17.45	17.46	+0.06
Full load GM	[ft]	5.98	6.41	+7.19
GM to Beam ratio	-	0.102	0.107	+4.90

Table 2.2. *Comparison of DDGx model and ASSET match runs for the DDG51.*

Underlined data implies numbers which are input to the model as independent parameters. The results are a little different than those presented in Appendix D, because here all calculations were performed by the Fortran self-balanced model automatically, and the results in appendix D were obtained manually, using the Mathcad model. It is not uncommon to obtain slight differences when using different numerical algorithms, even though they have been used to solve the exact same problem.

The comparison gives errors larger than 5% in three categories: hull depth at station 20, total required area and metacentric height. The error in D_{20} is irrelevant, because the DDGx model uses different sheer line, which is not calibrated with DDG51. The error in the required area prediction is not surprising, because as described in Section 2.10, ASSET seems to give unreasonable results in this field. The error in the result for GM is due to accumulation of small errors during the calculation per Equation (2.88). All the components in this equation have relative errors lower than 1%, but due to the low value of GM, the relative error is big. However, when the metacentric height is divided by the beam, to find the stability ratio, the error drops below 5%.

2.14 Experimental Results

Experimental results show that many of the solutions are non-feasible, generally due to stability problems of both kinds (see Section 2.12.6), and sustained speed violation, although other criteria are violated occasionally. Figure 2.14 reveals one of the outputs obtained from the Fortran model. It shows feasibility status of a randomly generated population of 30 individuals. The first six columns define the chromosome, the seventh gives the respective payload fraction, and the rest provide the status: an 'x' mark is written under a violated requirement. Each column is titled after the parameter associated with this constraint. Thus for example, the column 'Area' describes the deckhouse area constraint, 'kW' the electric plant limit, and so on. In case of negative resolved deckhouse volume, representing a too big hull structure, a '-' is assigned in the 'Vd' column. When an 'x' is written in this column, the maximum superstructure volume limit is exceeded. There are cases for which several criteria are violated. The stability constraint is the only criteria that is bounded from two directions. Hence, it is divided into two columns, one for unsatisfactory initial stability and the other for excessive stability.

```
 Cp   Cx    Cdl   Cbt   CD10   Crd    Fp    Area   kW   Vs   GM-   GM+   D10   Vd

.51  .85   82.6   3.3   10.4   .47  .1000                x
.51  .85   89.8   3.2   10.6   .78  .1000                x                 x
.56  .87   69.2   3.3   10.3   .05  .1056    x                                   -
.63  .88   73.4   2.9   11.4   .77  .1125                     x
.61  .71   72.4   3.4   11.6   .13  .1029    x                      x
.62  .80   66.1   3.5   10.7   .26  .1040    x                      x            -
.69  .85   80.0   3.0   10.1   .71  .1117                x    x
.52  .70   89.1   3.5   13.8   .05  .1015                x          x      x
.52  .77   87.9   3.2   11.7   .49  .1000                x          x
.58  .84   73.2   3.3   14.4   .11  .1158                           x
.59  .85   72.7   3.3   11.4   .26  .1094
```

.56	.86	73.2	3.2	10.7	.20	.1071	x				
.62	.82	77.2	3.4	11.7	.44	.1094				x	
.70	.84	60.1	2.9	13.2	.10	.1206			x		
.53	.70	67.3	3.4	11.1	.70	.0840	x	x		x	x
.66	.71	83.8	3.3	11.4	.78	.1032		x		x	x
.53	.81	66.2	3.2	12.1	.51	.1000				x	x
.64	.80	64.9	2.9	11.9	.71	.1085			x		x
.65	.84	89.0	3.6	11.7	.26	.1156		x		x	
.67	.76	78.2	2.8	10.9	.00	.1149					
.53	.85	76.2	2.8	11.4	.35	.1067		x			
.60	.77	82.2	3.6	12.8	.57	.1067		x		x	x
.59	.81	85.4	2.9	10.8	.01	.1133		x			
.52	.80	77.4	3.0	11.9	.09	.1036		x		x	
.52	.72	72.1	3.2	11.0	.53	.0912		x		x	x
.59	.87	82.5	3.3	12.7	.34	.1151		x		x	
.69	.84	84.3	3.0	10.7	.25	.1173		x	x		
.50	.84	63.3	3.4	11.6	.79	.0939		x		x	
.66	.71	79.8	3.3	11.2	.77	.1034		x		x	x
.62	.83	64.1	3.3	11.6	.78	.1055					

Figure 2.14. *Experimental feasibility status.*

It can be seen that only 3 chromosomes, 10% of the entire population, represent feasible solutions. Furthermore, 21 of 27 failures, almost 78%, involve a violation of the stability requirement. Additional experiments provided an even higher portion of stability-related non-feasibility. The remarkably interesting fact is that the majority of stability deviations are related to excessive stability, while the naval engineer typically deals with insufficient stability problems. It is found later, when optimum solutions are approached, that insufficient stability becomes the limiting stability constraint. More than 60% of the solutions suffer from sustained speed constraint violation, implying that fast hull shapes are also difficult to find.

Note that there are no deviations from the electrical plant requirement. In all runs performed this constraint is never violated. This implies that the electrical generators carried aboard DDGx are very powerful.

For all runs a 150MHz Pentium computer, with 16MB EDO RAM and 512K pipelined burst cache was used. In terms of running time, each chromosome evaluation takes 12.5 seconds on the average. Running time is driven by the deckhouse volume solver, or, specifically, on the proximity of the initial volume guess to the final result.

Chapter 3

Tune Up in the Unconstrained Solution Space

The system of constraints described in Chapter 2 limits the applicability of some possible solutions, thus decreasing the effective size of the solution space, because only feasible solutions are eligible, and the choice of the optimal ship is taken from among them. The difficulty is that neither the new boundaries nor the size of the reduced solution space are known; we can only say that it is included in the original domain. Figure 3.1 illustrates qualitatively the structure of the solution space.

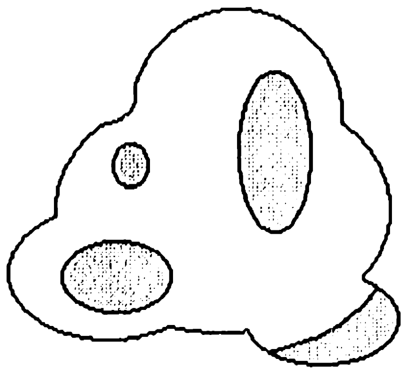


Figure 3.1. *Solution space.*

Grayed zones represent an environment of feasible solutions. As was learned experimentally in section 2.14, the allocation of real estate here is generous, and the real proportions are worse: likely less than 10% of the space provides feasible ships. There are some special methods to handle a constrained optimization, [6], [7] and [35] for example. However, before addressing this difficulty, it was decided to first test the search mechanism itself, in order to be confident with its correctness and performance. The search process needs many tune-ups to be effective, robust and reliable. This is equivalent to sensitivity research, in which system parameters are changed (here the verb tuned was used), the purpose is to find the best combination of them. Therefore, the first step performed is to ignore the non-feasible voids in the solution space while searching for the maximum payload fraction. The obtained data may not help in the final goal: the chances are that this overall maximum is non-feasible, due to the far bigger search area of such solutions. Nevertheless, a tuned algorithm facilitates the efficient search for the desired optimum.

This chapter defines the search parameters and describes the results of exploration under different combinations of them, without any constraints. The considerations in the decision of the best parameters depend on three outputs: average result, robustness and running time. Primarily, 'best' means that the algorithm finds the overall optimum in each run, and not a local one, after different population initializations (all of which are random). The measure is through the average optimum during a number of executions. In order to achieve a high average, the model must be robust. Even though it has started to search with different individuals, possibly poor, it is still expected to provide the same result. A robust model reduces the required amount of executions when trying to find the optimum. In a case that two exploration engines have similar performances, we can choose according to their convergence time.

A listing of the Fortran 90 source file is found in Appendix F. The file includes the converted Mathcad model shown in Appendix D and the evolutionary processes used during the research.

3.1 Search Parameters

Search parameters quantitatively express the features of an evolutionary exploration process, and are the means by which the search is governed. They consist of the population size *pop_size*, crossover probability P_C, mutation update probability P_{UP}, and the *selective pressure* P_S. The first three parameters were defined in Section 1.5; selective pressure will be discussed later in this chapter.

Although relatively simple, the DDGx self-balanced model is still very time intensive. It takes a 150MHz computer 12.5 seconds on the average to resolve one ship. An average generation therefore uses $12.5 \cdot$*pop_size* CPU seconds. This point of view suggests a reduced population size for relatively fast exploration. However, population size dictates the population diversity: the larger it is, the higher the chances that all required genetic material exists in it, thus avoiding premature convergence to a local optimum. This size provides the multi-directional search ability, and its advantage relative to other search methods, [9] and [10]. To ensure some feasible solutions at each generation, 30 individuals are used for the population size. A larger size takes too much time. A typical population size is on the order of 100 for similar applications, but it is possible to find lower sizes. The *span* of the population is defined as the generational standard deviation to average fitness ratio,

$$span(t) = \frac{\sqrt{\frac{1}{pop\_size - 1}\sum_{i=1}^{pop\_size}\left[fitness(x_i, t) - \overline{F}(t)\right]^2}}{\overline{F}(t)} \tag{3.1}$$

where t is the generation number, x_i a certain chromosome, and $fitness(x_i, t)$ is the fitness function value. The span measures the distribution of the fitness function around the average fitness at a specific generation. The higher the span, the higher the diversity of the population. At late stages of the exploration, for example, it is expected to decrease, because convergence is approached, and the population is assumed to be saturated with good individuals. [6] finds experimentally that an initial span of 0.1 gives a good trend between exploration (which is the same as diversity) and speed. Population size of 30 approximately matches this value. The maximum allowed number of generations, *gen_max*, is set to 100.

For the crossover probability the customary value of 0.25 is used. This means that, on the average, *pop_size*/4 individuals become parents. This has to be an even number, and thus a correction is added to the algorithm. If an odd number of chromosomes are selected, a coin is tossed. According to the outcome, one genotype is released, or an additional one is randomly selected.

The remaining two parameters, mutation update probability and selective pressure are not determined as constants, but rather are changed between executions, in order to obtain the best configuration of the search engine. The tune up uses these parameters.

For ship design, an accuracy of 4 decimal digits for the payload fraction, the fitness function, is satisfactory. In the event that the computer reports the same 4 digit maximum generational payload fraction for 10 successive generations, convergence is assumed even though generation *gen_max* has not yet been reached. There is some risk of premature convergence with this approach because there is always the possibility that the computer may find a better value given enough time. This risk increases as the crossover and mutation probabilities decrease, because then the chances to create new offspring, possibly better, are less. Due to the limitation on the precision of the final obtained optimum, the chances to observe a significantly better value are low.

3.2 Solution Space

Six-dimensional solution space boundaries are shown in Table 3.1 with the resolution of the search. The raised deck coefficient, C_{RD}, cannot take values higher than 1. Extreme values for the prismatic coefficient C_P and maximal section coefficient C_X are the traditional ones. C_P values also coincide with Gertler's reanalysis [20] for the residuary drag coefficient, see Section 2.5.2. ASSET limits the maximum section coefficient to 0.845, but, based on the experience of 13.412 projects, values to 0.90 are considered. Beam to draft ratio limits are taken from the original math model. The same values are recommended by ASSET. Values for the boundaries of the remaining two genes, the displacement to length and length to depth ratios are chosen to contain DDG51 actual numbers. At $C_{\Delta L}$ values lower than 60 lton/ft^3, weight convergence is not observed. The initialization of the population is done by the expression

$$C = C_{\min} + \frac{C_{\max} - C_{\min}}{N} \cdot RAN\# \tag{3.2}$$

where *RAN#* is a randomly generated integer number, with a uniform distribution, from the domain [0,*N*].

Gene	Minimal value	Maximal value	Divisions (N)
C_P	0.50	0.70	20
C_X	0.70	0.90	20
$C_{\Delta L}$ [lton/ft^3]	60	90	15
C_{BT}	2.8	3.7	9
C_{D10}	10	15	10
C_{RD}	0	0.8	20

Table 3.1. *Solution space boundaries and search resolution.*

Table 3.1 dictates a discrete search which is not really necessary in a floating point number representation. However, the specified precision of the results decreases the number of optional solutions, thus decreasing the running time. The selected resolution also provides a justification for the usage of evolutionary strategies. The total time required to conduct a complete exhaustive search through the entire space can be computed by multiplying the overall number of combinations by the average running time,

$$20^3 \cdot 15 \cdot 10 \cdot 9 \cdot runs \cdot 12.5 \cdot \frac{sec}{run} \cdot \frac{1}{3600} \cdot \frac{hour}{\sec} \cdot \frac{1}{24} \cdot \frac{day}{hour} \cdot \frac{1}{365} \cdot \frac{year}{day} = 4.28\ years$$

Thus more than 4 years and 3 months are required to cover all the possible solutions.

As stated in Section 1.3, the resolution of the search is driven by the desired accuracy of results. Mathematically written, it is expressed by Equation (1.1),

$$N_i = (b_i - a_i) \cdot 10^k \tag{1.1}$$

where N_i is the number of divisions in the domain $[a_i, b_i]$. The most precise genes are the first two, C_P and C_X, with an accuracy of 0.01, due to the model's sensitivity to these parameters. Other parameters are given lower accuracy. After a maximum value is found, it is possible to increase the resolution. This is done later in Chapter 5.

A binary representation requires 27 bits in each chromosome. For the reasons listed in Section 1.3, floating point number representation is adopted here. The floating number chromosome is a six-dimensional vector.

3.3 Classical Search

It was natural to start with the classical exploration process, as developed by Holland [5] at the University of Michigan. This is the most basic strategy, and its details are described and discussed in Chapter 1. This is a pure genetic algorithm. Several runs with different mutation update probabilities were performed. None of them converged. Convergence chances increase as the mutation probability decreases, because the search is narrower and very conservative in creating new solutions. Figure 3.2 provides a graphic representation of the process under a 0.1 mutation update probability.

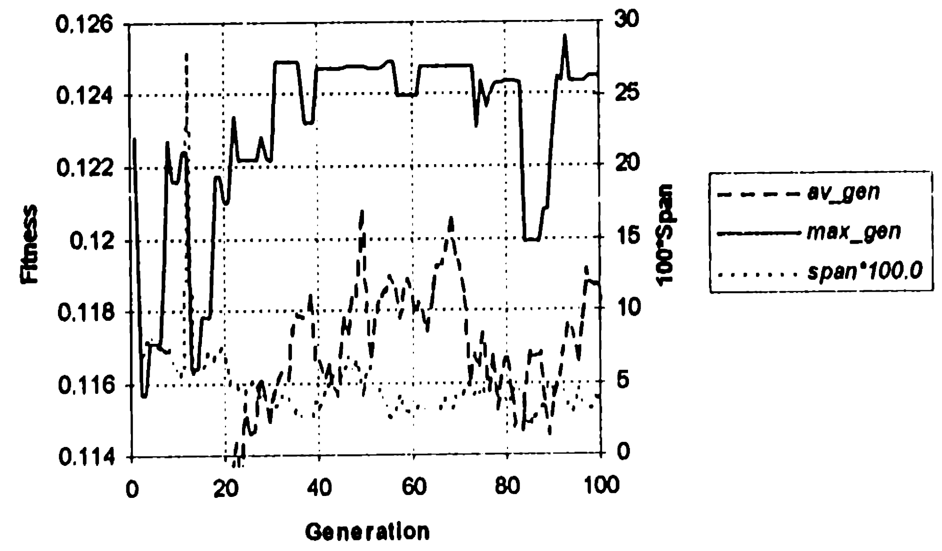


Figure 3.2. *Classical genetic algorithm results with P_{UP}=0.1.*

The uppermost curve, designated in the legend by *max_gen*, describes the highest payload fraction individual in each generation. The best ever found value, 0.1256, was at generation 92. The curve denoted by *av_gen* represents the average fitness of all chromosomes at every generation. The third line draws the behavior of the span, defined in Equation (3.1). The exploration is virtually random; there is no evidence of sustained improvement of the maximal generational payload fraction as the process advances.

This failure is a result of the characteristics of the payload fraction function. The distribution of a randomly initialized population performance is shown in Figure 3.3. The solid horizontal line denotes the average fitness of the population. This is a typical population - the average is not high, and none of the individuals has a payload fraction higher than 0.119.

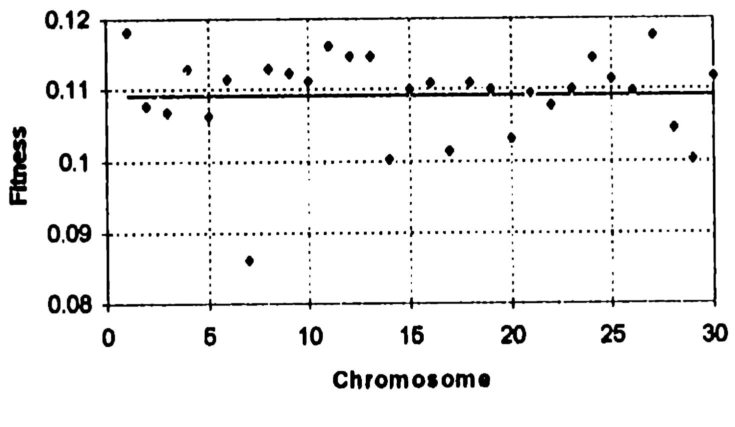


Figure 3.3. *Payload fraction of a randomly initialized population.*

Except for one bad chromosome, the seventh, the relative differences are within ±10% about the average. In Section 1.4 the selection scheme is discussed, in which each chromosome is given a probability of survival,

$$P_i = \frac{fitness(x_i)}{\sum_{i=1}^{pop\_size} fitness(x_i)} \tag{1.4}$$

Rephrasing Equation (1.4), and writing it as a function of the generational average fitness,

$$P_i = \frac{fitness(x_i)}{\overline{F}(t) \cdot pop\_size} \tag{3.3}$$

This reveals that, given a distribution similar to that of Figure 3.3, the probabilities of survival have close values. In GA terminology, the problem is referred to as a *scaling problem*. Figure 3.4 best illustrates this using the resultant roulette wheel. As in Figure 1.2, the probabilities are sorted clockwise in a decreasing order. According to classical selection methodology, based on the larger amounts of wheel real estate claimed by the more fit individuals, they have higher probability to survive and reproduce. However, in the case of Figure 3.4, the slots are similar in area - there is no distinct difference between the slice associated with the poorest solution and the best one. These are intentionally located one beside each other, at the top of the wheel.

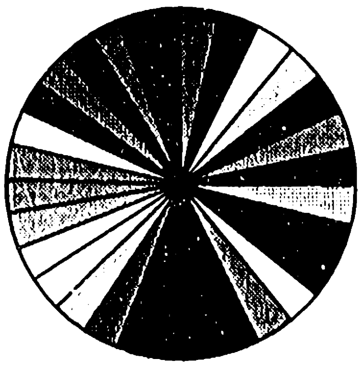


Figure 3.4. *Roulette wheel in a classical search.*

Consequently, the problem is the seemingly inappropriate selection algorithm. This is a classic case of poor exploitation of the better genotypes. Instead of using the good genetic material, the computer ignores their promising performance, and actually conducts a random search across the solution space. A short Fortran code was used to count the times each chromosome is selected. The experimental result for the roulette wheel of Figure 3.4 is shown in Figure 3.5. Again, the chromosomes are ranked according to their fitness from right to left, in descending order.

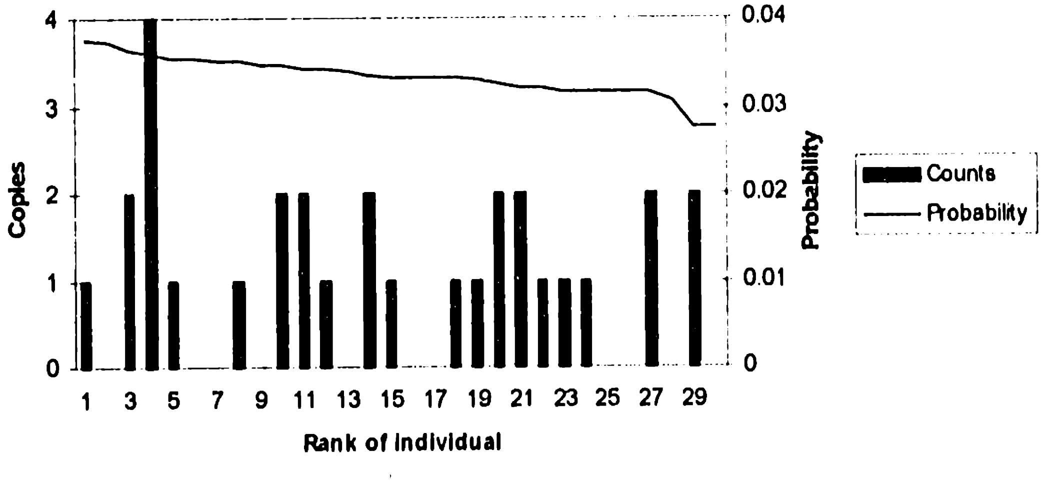


Figure 3.5. *Experimental reproduction in a classical selection.*

High fitness solutions are not selected more frequently, and bad solutions are given more copies. The graph also shows the flatness of the probability curve. A new measure, the *center of selection*, an equivalent to the center of gravity in mechanical systems, is introduced,

$$CS = pop\_size - \frac{\sum_{i=1}^{pop\_size} counts(x_i)\cdot(pop\_size - i)}{pop\_size} \qquad (3.4)$$

The center of selection is the expected position (in a ranked list, where the best individual has the first position) of a randomly picked chromosome. That is, it gives the center of gravity of the bars in Figure 3.5. This parameter provides a statistical measure of exploitation: the lower the location, the more exploitative is the algorithm. $counts(x_i)$ is the number of times that a chromosome x_i has been copied. For example, an entirely random selection has a center of selection of $(pop\_size+1)/2$, 15.5 in this research; an elitist model, which selects only the best individual, has a center of 1. Quick computation shows that the center of selection of Figure 3.5 is 14.43, virtually a random selection.

3.4 Rank-Based Selection and Selection Pressure

In order to enable exploitation, to an appropriate extent, the program must distinguish between fit and poor individuals, and sample them in amounts that agree with selection theory. As demonstrated in the last section, particularly in Figure 3.3, the payload fraction function is too flat. Plotted with its six degrees of freedom, it is nearly a flat surface with very distinctive maximum points rising like high mountains. This is the reason that a random initialization gives modest (in payload fraction terms) ships - most of the surface is flat, and it is hard to find the mountains.

There are many methods to improve the selection procedure. Michalewicz [6] dedicates a special chapter[1] to such problems, and relates them to the sampling mechanism and the characteristics of the evaluation function. He is concerned with premature convergence, vice the current problem of no convergence. Goldberg [7][2] suggests to use a *linear scaling*, in which the fitness of all chromosomes is scaled by the law

$$fitness'(x_i) = a \cdot fitness(x_i) + b \qquad (3.5)$$

where a and b are problem dependent parameters. This scaling law removes the congestion of fitness values around their average, and helps emphasize the bulges on the surface. A similar correction is attained using the *power law scaling*,

$$fitness'(x_i) = fitness^k(x_i) \qquad (3.6)$$

with k as a system parameter.

[1] Chapter 4: Selected Topics.
[2] pp. 122-124.

A different method, with the same effect, is to sort the chromosomes in descending order, according to their fitness, and to assign each individual a probability of survival based on its location i. The best chromosome is ranked 1, and the poorest gets the rank *pop_size*. The selection process is then rank-based, and the probability function is a geometric series,

$$P_i = A_1 \cdot P_S^{\,i-1} \tag{3.7}$$

Here i goes between 1 and *pop_size*, A_1 is a constant, and the *selection pressure*, P_S, is the quotient of the series. P is a regressive series, otherwise poor solutions will get higher probabilities than good ones. Hence, the quotient of the series ,P_S, takes values between 0 and 1. The sum of all probabilities has to be 1, so A_1 is a dependent variable,

$$A_1 = \frac{1 - P_S}{1 - P_S^{\,pop\_size}} \tag{3.8}$$

The closer the value of P_S to unity, the more uniform is the probability. At the limit of P_S=1, the series does not exist, and all individuals get the same probability of survival,

$$P_i(P_S = 1) = \frac{1}{pop\_size} \tag{3.9}$$

The selection is then totally random. As P_S approaches zero, A_1 approaches unity, while all other selections are negated. P_S dictates the selection: the closer it is to zero, the more elitist is the selection; the closer it is to unity, the more random is the selection. Figure 3.6 shows the probability of survival as a function of selection pressures and the rank of the chromosome.

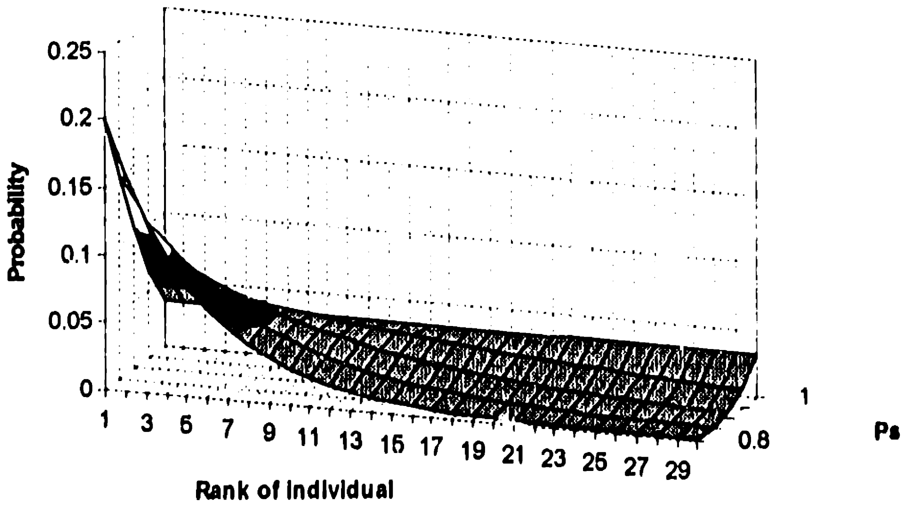


Figure 3.6. *Probability of survival under different selection pressures.*

The surface is a plot of Equation (3.7), with the first term computed from (3.8). A selection pressure of 0.8 is very elitist, and the probability of the lower half of the population is approximately zero. As the pressure approaches unity, the cross section of the plane becomes a horizontal line, assigning identical chances of breeding to all the population, without considering their respective performance.

Figure 3.7 shows experimental selection results under variable pressures. At low selection pressures, the columns tend to crowd to the left, and become higher; no relatively poor solutions survive. When the pressure increases, the distribution and height of the columns become uniform.

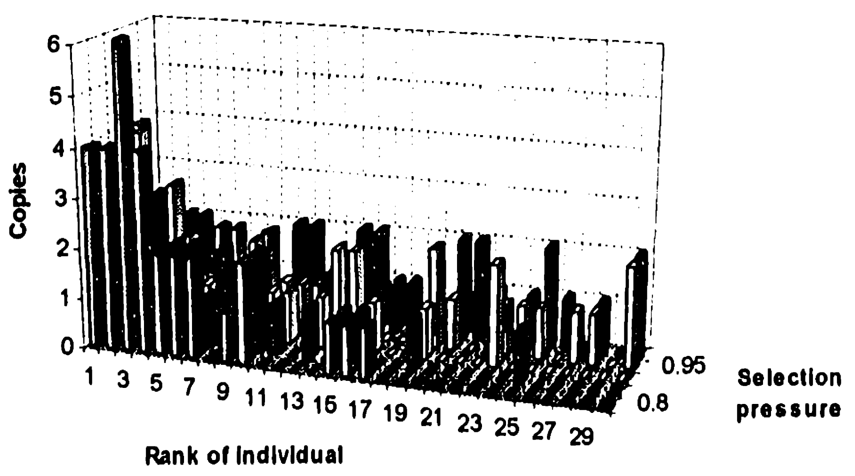


Figure 3.7. *Experimental distribution of copied chromosomes under different selection pressures.*

Since Equation (3.7) represents a deterministic survival probability function, the theoretical location of the center of selection can be computed,

$$CS = \sum_{i=1}^{pop\_size} P_i \cdot i \tag{3.10}$$

Equation (3.10) is derived using probability theory. The theoretical location, with some experimental results, are shown in Figure 3.8. The experimental points match the curve nicely. The real curve can take only integer values, so it actually has steps. The curve of Figure 3.8 is the pure mathematical result, with no truncations or rounds off. The curve reveals the greatest advantage of rank-based selection methods: the exploitation is both predictable and controllable, with minimum effort. The exploration and exploitation struggle is in the hands of the user. For space-wide exploration, the pressure is increased; for tight exploitation, it is decreased[3].

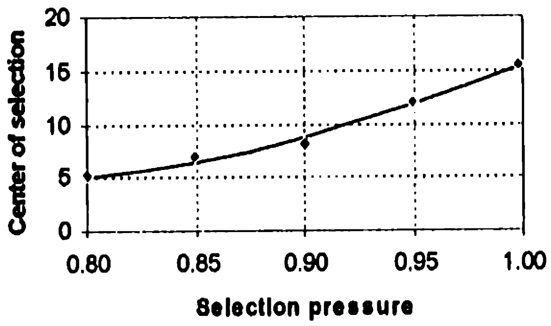


Figure 3.8. *Center of selection in rank-based selection.*

The rank-based procedures do suffer from several disadvantages. They disregard similarities of solutions: relatively close fitness values do not get close probabilities to survive and breed. Secondly, these methods violate the schema theorem described in Section 1.6.2, since the whole idea of showing that higher than average individuals get exponentially increasing number of copies relies on the proportional probability of survival. The advantages of total control they give and the guarantied cancellation of all scaling problems, outwage these minor drawbacks, and this approach is used in subsequent algorithms[4].

3.5 System Parameters Tune Up

Assuming that once scaling problems are eliminated, convergence can be reached, the next goal is to avoid premature convergence. In the literature, premature convergence is considered the main problem of evolutionary programs, and many solutions have been suggested; a brief description with rich bibliography is given in [6]. At this stage the only departures from traditional genetic algorithms are the float number chromosome representation and the rank-based selection. Before starting to add some enhanced genetic operators, experiments with the existing model show that even with a rank- based selection, high mutation rates lead to divergence. There is no surprise here: as the probability to mutate increases, the concentration is on search rather on exploitation; very high rates create wild exploration, almost random. A second expected result is the

[3] There is a basic confusion in the definition of selection pressure: an increased pressure loosens the selection, and decreased pressure tights the procedure. I have decided to keep this name only because the term is widely used in evolutionary strategies. However, the implication should be imprinted in mind.
[4] There is the biggest advantage of this method, which is discussed in Chapter 5, when the search is conducted in the constrained space. The method allows big penalties applied to non-feasible solutions.

premature convergence at low selection pressures[5]. Pressures lower than 0.9 are virtually ineffective. In such runs the algorithm locks on a certain super-individual and converges rapidly to a local maximum. Thus, tune up was performed at relatively low mutation update probabilities (of 0.1, 0.2 and 0.3) and relatively high selection pressures (0.94, 0.96 and 0.98). 10 runs were executed per each combination of P_{UP} and P_S, for which the resolved maximal payload fraction and the respective convergence time (in generations) are given in Tables 3.2 and 3.3.

P_{UP}	0.1			0.2			0.3		
P_S	0.94	0.96	0.98	0.94	0.96	0.98	0.94	0.96	0.98
1	0.1278	0.1269	0.1244	0.1254	0.1275	0.1269	0.1272	0.1273	0.1262
2	0.1275	0.1269	0.1268	0.1277	0.1265	0.1261	0.1264	0.1263	0.1272
3	0.1266	0.1260	0.1255	0.1273	0.1269	0.1269	0.1278	0.1260	0.1277
4	0.1264	0.1271	0.1258	0.1266	0.1267	0.1281	0.1271	0.1278	0.1260
5	0.1261	0.1260	0.1239	0.1270	0.1259	0.1265	0.1272	0.1272	0.1277
6	0.1256	0.1269	0.1225	0.1264	0.1273	0.1265	0.1280	0.1275	0.1255
7	0.1273	0.1269	0.1253	0.1266	0.1273	0.1267	0.1261	0.1274	
8	0.1268	0.1272	0.1233	0.1273	0.1272	0.1269	0.1266	0.1271	
9	0.1280	0.1277	0.1251	0.1278	0.1268	0.1262	0.1278	0.1267	
10	0.1263	0.1276	0.1252	0.1281	0.1259	0.1268	0.1273	0.1278	
Average	0.1268	0.1269	0.1248	0.1270	0.1268	0.1268	0.1272	0.1271	0.1267

Table 3.2. *Maximal payload fraction.*

P_{UP}	0.1			0.2			0.3		
P_S	0.94	0.96	0.98	0.94	0.96	0.98	0.94	0.96	0.98
1	55	40	48	31	42	63	55	24	100
2	60	34	37	63	32	44	40	35	100
3	34	55	48	42	40	71	41	35	100
4	23	30	46	26	63	68	50	63	100
5	22	15	20	37	24	46	26	43	100
6	20	40	13	33	33	78	36	35	100
7	34	23	41	21	30	35	18	72	
8	38	27	12	37	61	56	30	61	
9	26	58	19	34	27	37	36	43	
10	24	28	59	51	42	72	51	100	
Average	33.6	35.0	34.3	37.5	39.4	57.0	38.3	51.1	100.0

Table 3.3. *Generations for convergence.*

The overall maximum payload fraction obtained in these 86 runs is 0.1281, twice. For the right hand column no convergence is reached after 100 generations due to the high

[5] Note again to the basic confusion in the definition of selective pressure.

pressure and mutation probability, so only six runs were performed. The best algorithm is the one that gives the highest average fitness. This happens for a mutation probability of 0.3 and selection pressure of 0.94. The trends in Table 3.3 are as expected: as the pressure and the probability of mutation are increased, the concentration is on exploration, and the average time to attain convergence increases. A typical result is shown in Figure 3.9.

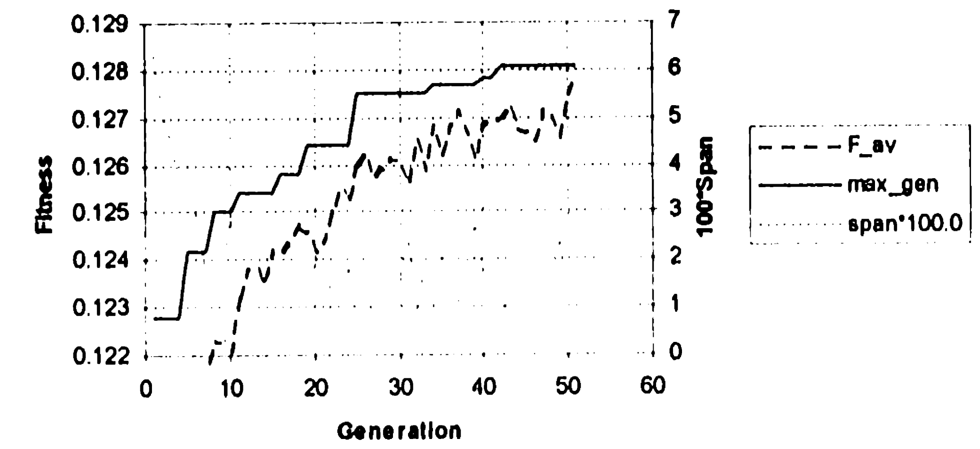


Figure 3.9. *Rank- based algorithm results, P_{MP}=0.2, P_S=0.94.*

In this example the algorithm did find the highest value of 0.1281. Note the differences from the behavior of Figure 3.2: first and most significant, the average generational fitness agrees with the survival of the fittest principle and evolutionary theory - there is clear improvement as the procedure progresses. The improvement in the maximal generational value is continuous, and the span decreases towards convergence, because the population becomes saturated with good individuals.

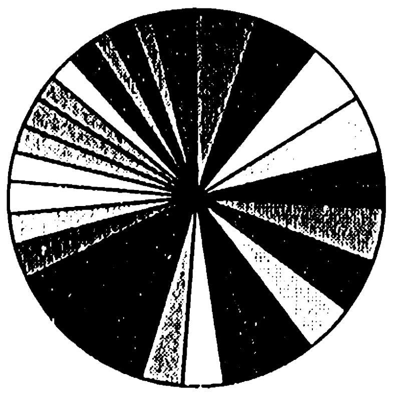


Figure 3.10. *Roulette wheel in a rank-based algorithm.*

Figures 3.10 and 3.11 show the roulette wheel and reproduction results at a certain generation, for a pressure of 0.96, as opposed to Figures 3.4 and 3.5. With the ranking, high performance solutions get higher chances to survive. In Figure 3.11, the center of selection is 12.37.

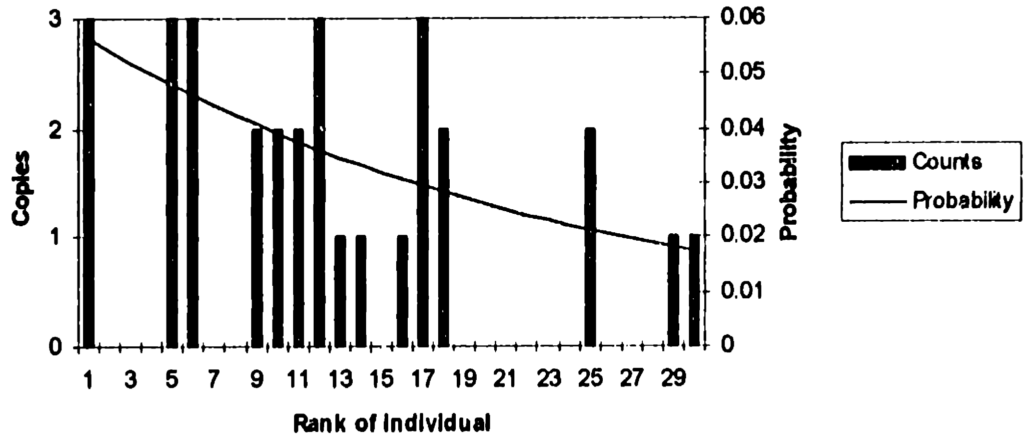


Figure 3.11. *Experimental reproduction in a rank-based selection algorithm.*

A typical chromosome evaluation takes 10 to 15 seconds; each generation lasts more than 6 minutes, and 50 generations take about 5 hours. However, as the process goes on, there is convergence to the best solutions, and some chromosomes in the same generation are identical. Also, as the probabilities for crossover and mutation decrease, and/or the sampling mechanism becomes more elitist, the chance of having equal genotypes increases. Therefore, an accelerating algorithm is added, which first checks whether a solution has already been evaluated in the population, and if so, just duplicates the already available solution and proceeds. The check for identical chromosome is performed only at the current processed generation, and hence the chances of duplication increase as the number of evaluated chromosomes increases towards *pop_size*. For example, in the second individual evaluation only one chromosome is checked. In the third individual the first and second individuals are checked and so on. Using the accelerator has proven to be very effective. Figure 3.12 displays the number of duplicated solutions for a certain run. The average number of duplicated individuals is 6.36, thus a time saving of more than 20% is gained.

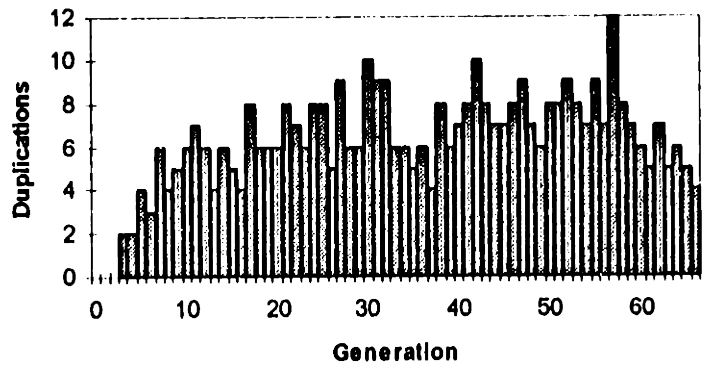


Figure 3.12. *Number of duplicated chromosomes per generation, P_{cp}=0.3.*

The overall best ship violates three constraints: initial stability, depth and deckhouse volume. In Chapter 4 several ships with the same maximum payload fraction of 0.1281 are found. Figure 3.13 provides the technical data of one of them.

```
Main Parameters ...

  Cp                                          .700
  Cx                                          .900
  Cdl                                         75.93 [lton/ft^3]
  Cbt                                        2.800
  CD10                                       15.00
  Crd                                         .000

  Main Dimensions on Waterline ...

  Underwater volume                           2.520E+05 [ft^3]
  Length on waterline                        456.10 [ft]
  Beam                                        49.56 [ft]
  Draft                                       17.70 [ft]
  D10                                         30.41 [ft]

  Speed and Resistance ...

  Wetted area                                28860.7 [ft^2]
  Sustained speed                            30.48 [knt]
  Maximal speed                              32.48 [knt]
  Endurance shaft horsepower                 16493.0 [hp]

  Space Available ...

  D0                                         46.96 [ft]
  D10                                        30.41 [ft]
  D20                                        34.91 [ft]
  Total hull volume                           597002.3 [ft^3]

  Electrical Loads ...

  Winter cruise electrical load              3267.7 [kW]
  Marginal winter cruise electrical load     3300.3 [kW]
  24 hours average electrical load           2200.4 [kW]
  Power required per generator               1833.5 [kW]
  Auxiliary machinery rooms volume           54508.8 [ft^3]
```

```
Tankage ...

Fuel weight                              1213.7 [lton]
Fuel tanks total volume                  54986.6 [ft^3]
Ballast tank volume                      10447.5 [ft^3]
Total tanks volume                       72885.9 [ft^3]

Weight and Center of gravity ...

Lightweight                               5470.6 [lton]
Vertical CG of lightship                 20.94 [ft]
Full load displacement                    7176.0 [lton]
Vertical CG at full load                 19.54 [ft]
Payload fraction                          .128

Area/volume Balance ...

Required hull area                       48925.1 [ft^2]
Available hull area                      31049.5 [ft^2]
Required hull volume                     521541.3 [ft^3]
Available hull volume                    330987.7 [ft^3]
Required deckhouse area                  10042.1 [ft^2]
Available deckhouse area                 27701.3 [ft^2]
Required deckhouse volume                107049.2 [ft^3]
Available deckhouse volume               295295.6 [ft^3]
Total required area                      58967.2 [ft^2]
Total available area                     58750.8 [ft^2]
Total required volume                    628590.5 [ft^3]
Total available volume                   626283.3 [ft^3]

Initial Stability .....

KB                                        9.85 [ft]
BM                                       12.96 [ft]
Metacentric height                        3.27 [ft]
GM to B ratio                             .066

Deckhouse volume                         296949.4 [ft^3]

Balance/Feasibility Status ...

Category                            Required/Minimal    Available
Deckhouse area [ft^3]                    10042.1          27701.3
Electric plant [kW]                      1833.5           2500.0
Sustained speed                          29.96            30.48
Initial stability                         .090            .0660
                                          .135
Depth [ft]                               32.66            30.41
Maximal deckhouse volume [ft^3]          293917.4         296949.4
Payload fraction                          .075            .1281
```

Figure 3.13. *Self balanced DDGx unconstrained best model output.*

All population initializations are random. It can be seen from Table 3.2 that the algorithm, for any combination of the search parameters, is not robust. The variability of the results is quite high, and the averages of fitness function maximum values differ from the best ever value of 0.1281 in the third decimal place. The payload fraction range is approximately between 0.085 (the lowest value in Figure 3.3) and 0.1281 (the highest value in Table 3.2). The best parameter combination of P_{UF}=0.3 and P_S=0.94, giving the highest average fitness of 0.1272, is still below the optimum by 2.1%. The best combination therefore suggests an accuracy of 97.9%. Note also that this combination offers good speed, 38 generations to converge on the average.

In some runs, the final result was lower than the best-ever result found during the exploration. An example is given in Figure 3.14.

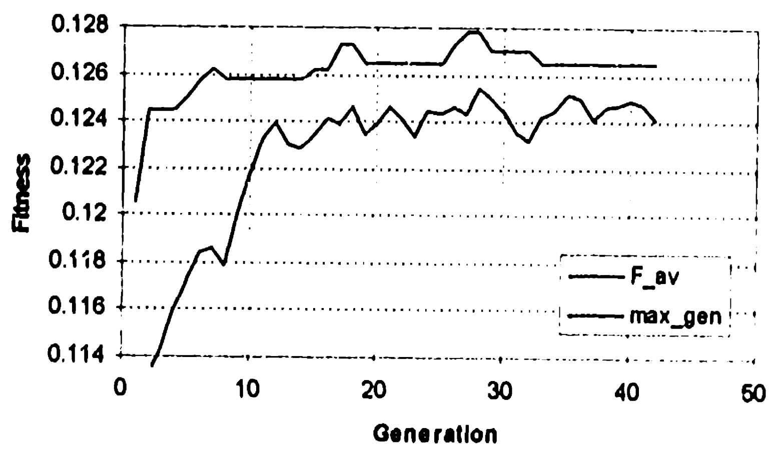


Figure 3.14. *Convergence to a local maximum.*

Here two problems exist: the best ever solution has a 0.1278 payload fraction (and not the optimal 0.1281), and the final reported value is 0.1265. This is a result of the stochastic errors in the sampling mechanism, and it occurs more and more as the probability of mutation increases - the computer mutates the generational best chromosome so it actually forgets it and loses the good genetic information it bears. This is the reason for storing the best ever value in addition to the generational value. Some elitist models enforce the selection of the best ever value in order to prevent this phenomenon, but this approach is not adopted here, in order to prevent premature convergence. The modified genetic operators described in Chapter 4 solves the problem of Figure 3.14.

Chapter 4

Enhanced Evolutionary Strategies in the Unconstrained Solution Space

In Chapter 3 the foundations for the genetic search are presented, and values for system parameters which gave the best results are identified. Although the selected configuration accuracy is 97.9%, the highest average payload fraction differs from the best ever value in the third decimal digit: 0.1272 versus an optimum of 0.1281. Except for the usage of a rank-based selection mechanism, and the floating point number representation, this is still the basic genetic algorithm search procedure suggested by Holland [7]. The selected configuration of Chapter 3 : .eferred as the *datum* model. Due to the increasing interest in evolutionary strategies, and the availability of parallel computing systems, a lot of academic effort was made during the last two decades to develop new methods in this field. This chapter introduces some of the improved strategies incorporated in the search model in order to increase the robustness of the datum system. Due to the intensity and complexity in the evaluation of the DDGx model, an increased robustness means increased confidence in the results - and, in the bottom line, a reduced number of required executions. For example, return to Table 3.2. Through all 86 runs performed, only two have identified the overall maximum payload fraction. Note also that the selected parameter combination - the datum - does not report these values at all. By using the following new operators the model becomes an evolutionary program. During all runs, when applicable, the same system parameters as in the datum optimizer are used, except one: the convergence criterion *n_converge* is increased to 15 successive generations instead of 10; the algorithm is made more "patient". Each new operator is evaluated alone, to asses its contribution, and then several combinations are examined. The search is still evaluated without the physical constraints of the system because it is easier to handle one problem at a time: after the premature convergence problem is eliminated, the model can be used with more confidence in any environment.

4.1 Non-Uniform Selection Pressure

The fundamental cause in large population systems for premature convergence is the ignorance of some vital genetic information. While small populations could possibly lack required genetic material, large populations, together with an efficient search engine, insure that the exploration has the necessary genetic data. The only problem is how to refine and extract it. Premature convergence happens when one or more super-individuals, with superior fitness, dominate the population. These super-individuals are copied many times by the selection operator, and very fast the population is saturated while other data is destroyed. In such cases the computer concentrates on exploitation and forgets about exploration. In order to control the reproduction process, and to bound the exploitation, DeJong [38] suggests to limit the number of copies a chromosome can have. From a different point of view, the center of gravity in the search must move from exploration at the beginning, to exploitation at the end: At the beginning, the exploration is almost random, allowing the evaluation of many options, until all vital information is collected. This prevents the influence of super-individuals. Later, when the algorithm has found the promising regions, there is less interest in new solutions, and the computer concentrates only on the good chromosomes. The shift from exploration to exploitation is expressed by means of the center of selection, and is controlled by the selection pressure: high pressure is associated with exploration, low pressure with exploitation[1]. Instead of using a constant pressure, a *non-uniform selection pressure* is exerted. In Chapter 3 it is concluded that the most effective selection pressure is 0.94. Keeping this as the average pressure, the specific pressure is changed from 0.98 at the initialization to 0.92 at generation *gen_max* linearly,

$$P_s = -0.06 \cdot \frac{gen}{gen\_max} + 0.98 \tag{4.1}$$

Experiments show that lower pressures are too elitist, and give no improvement. Table 4.1 displays experimental results obtained using the non-uniform selection pressure. All other parameters are as selected at the end of Chapter 3. *gen* represents the number of generations before convergence. The right column gives the average of ten runs.

Run	1	2	3	4	5	6	7	8	9	10	Ave
F_P	0.1281	0.1273	0.1274	0.1281	0.1268	0.1276	0.1271	0.1272	0.1280	0.1278	0.1275
gen	41	53	63	100	40	79	100	83	100	84	74.3

Table 4.1. *Experimental results with non-uniform selection pressure.*

[1] I promise that this is the last time: pay attention to the fundamental confusion in the definition of selection pressure.

The results are better than before: an average of 0.1275 instead of 0.1272, implying an accuracy of 98.6% as opposed to the former precision of 97.9%. The process does take more time, but this is the price of avoiding premature convergence.

4.2 Fine Local Tuning with Non-Uniform Mutation Operator

The mutation operator creates new solutions by altering one randomly selected gene, to a random extent (but within the predefined gene domain). The operator can provide some important missing genetic material, critical for the optimal solution. Assuming continuity, it also provides a new individual in the vicinity of the mutated one. It may have a higher fitness. Unfortunately, it might also offset the algorithm from a promising regime if the change is too extreme. Such a behavior is acceptable and even desired in early generations because it provides extensive exploration capability, and a wide search ensures that the algorithm has checked many options before concentrating on a certain one. At progressive stages, where the algorithm approaches the best solution and starts to converge, it distracts and can destroy the good genetic material. An example of this situation was illustrated in Figure 3.14. The extent to which a gene is changed should be generation dependent: as the generation number increases, the mutation updates a gene nearer to its original value. The operator is then a *non-uniform mutation* operator, because the change is not uniform throughout the whole process. The mechanism is referred to as *fine local tuning*. Mathematically speaking, if a gene x_i is bounded in the domain $[a_i, b_i]$, than its mutated value is given by,

$$x_i' = \begin{cases} x_i + \Delta(gen, b_i - x_i) & \textit{if a random digit is } 0 \\ x_i - \Delta(gen, x_i - a_i) & \textit{if a random digit is } 1 \end{cases} \tag{4.2}$$

where the effective branch in Equation (4.2) is selected by tossing a coin: the first mutates to a higher value, the later to a lower value. *gen* is the generation number, and,

$$\Delta(gen, x) = x \cdot \left[1 - r^{\left(1 - \frac{gen}{gen\_max}\right)^{b}} \right] \tag{4.3}$$

where r is a uniformly distributed random number from the range [0, 1], *gen_max* is the maximum allowed number of generations, and b a parameter determining the degree of non-uniformity. The function Δ determines how much a gene is mutated, and is displayed in Figure 4.1, for b=2. Each curve represents different progresses in the search, expressed by the non-dimensional time *gen*/*gen_max*, of 0.1 through 0.9. As the process advances, the possibility of large changes decreases; as b increases, the decay of the graphs becomes more drastic, reducing further the chances of serious offsets.

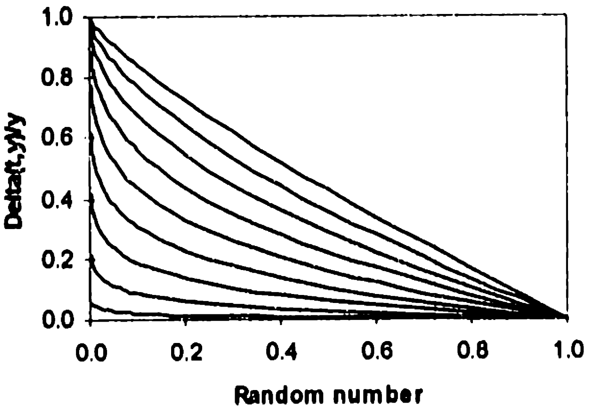


Figure 4.1. *The non-uniform mutation Delta function, for different gen/gen_max ratios, b=2.*

The influence of the parameter *b* can be seen in Figure 4.2. It shows curves for *b* varying between 1 and 5, for a constant non-dimensional time of *gen/gen_max*=0.3.

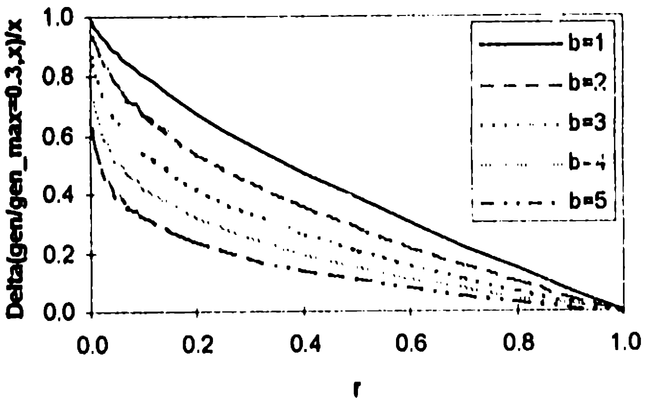


Figure 4.2. *Influence of the parameter b on the Delta mutation function, gen/gen_max=0.3.*

When use is made of this operator, the search mechanism is very pedantic, and usually does not lose the best result, as happened in Figure 3.14. The exploration at these stages is so local, that the algorithm recovers even if the best value has been forgotten for a certain period. Table 4.2 provides experimental results with the non-uniform mutation operator. The parameter *b* is set to 2. The performance with this operator is even better, with an accuracy of more than 99.5%. The speed of convergence is also higher relative to the last section. Clearly, this is a better operator, with the free advantage of shorter running time.

Run	1	2	3	4	5	6	7	8	9	10	Ave
F_P	0.1281	0.1273	0.1278	0.1275	0.1281	0.1281	0.1280	0.1279	0.1280	0.1281	0.1279
gen	34	59	48	49	100	52	78	58	70	59	60.7

Table 4.2. *Experimental results with non-uniform mutation operator, b=2.*

Evidently, the accuracy requested - before using this operator - for several genes, C_{M}, for example, was not enough, and the search was too coarse. The non-uniform mutation operator fixes this because it assigns the genes mid-values as well. Apparently there is big sensitivity to the gene values. A minor disadvantage of the non-discrete search, on the other hand, is that the chances to duplicate solutions in the next generation decrease. Consequently, runs in this case are longer in total CPU time (but not in convergence time).

4.3 Baker's Selection Mechanism

High selection pressure operated at early search stages reduces the concentration on super-individuals. However, to conduct an entirely random search at the beginning, by using a very high pressure, say 0.99 and up, is inefficient because we still want to be able to have some distinction between good and poor chromosomes, and to learn something from the population, until the important genetic material is identified. Baker [37] presented a stochastic universal sampling method, in which the whole population is exploited, thus collecting genetic data generation-wide, under predefined probabilities of selection. The model uses the same roulette wheel, with slot sizes allocated according to the relative fitness, but incorporates *pop_size* equally spaced markers, and only one wheel spin. The numerous uniformly distributed markers guarantee diverse selection, still depending on the slot area: high fitness individuals still have higher chance to get more copies. Figure 4.3 illustrates an experimental result of Baker's selection filter.

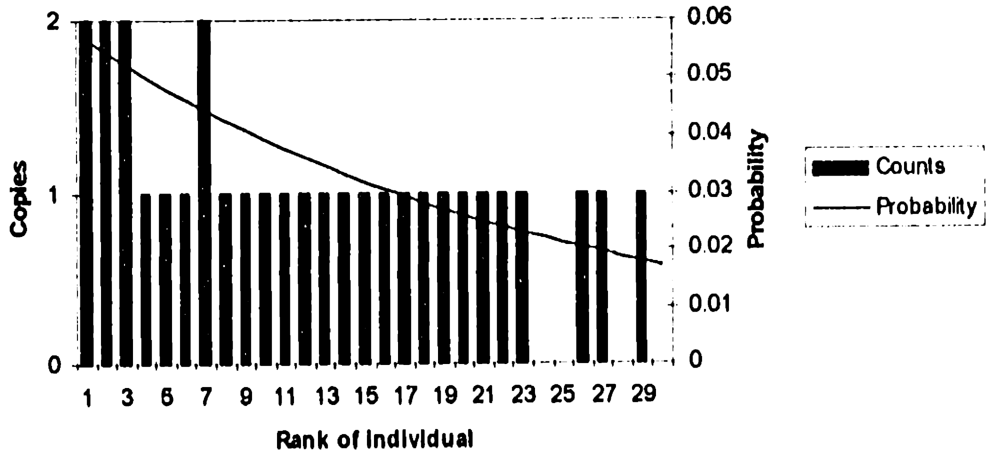


Figure 4.3. *Experimental reproduction according to Baker's selection algorithm.*

In this case the same selection pressure of Figure 3.11 is used, for comparison purposes. Except for two inferior chromosomes, all solutions have been selected; the best three (plus the seventh) individuals received double copies, to reflect their superiority. The improved selection mechanism can be appreciated now: it obeys the selection theory, because good individuals get more chances to survive and breed, but also explores the genetic data stored in the whole population. According to [6], this method was adopted by many researchers as a standard. The experimental center of selection in this example is 12.37, the same as in the regular selection of Figure 3.11, as it should be. In a rank-based roulette wheel construction, it depends on the selection pressure only,

$$CS = \sum_{i=1}^{pop\_size} P_i \cdot i \tag{3.10}$$

Because the probabilities are functions of P_S,

$$P_i = A_1 \cdot P_S^{\,i-1} \tag{3.7}$$

Table 4.3 shows experimental results with Baker selection. The average predicted payload fraction is the same as in the datum model, but inferior relative to former trials. The good news is that the speed of convergence is significantly higher than before: the algorithm collects all vital genetic information very effectively. However, when it has the data, it does not have the ability to complete the exploration without premature convergence, as in the datum model.

Run	1	2	3	4	5	6	7	8	9	10	Ave
F_P	0.1259	0.1280	0.1281	0.1270	0.1273	0.1277	0.1268	0.1275	0.1260	0.1281	0.1272
gen	37	28	100	28	33	75	38	50	40	58	48.7

Table 4.3. *Experimental results with Baker's selection method.*

4.4 High Order Crossover Operator

Another approach to improve the search mechanism refers to the basis of the theory of genetic algorithms. As shown in Section 1.6, the exploration is conducted via the concatenation of building blocks, groups of fit genes that are brought together towards the fittest individual. The schema theorem states that short, low order, above average schemata receive exponentially increasing samplings in subsequent generations, so only a portion of the available schemata is exploited. For instance, suppose that the high fitness schema S=(110* * * * *0101) is evaluated. It is destroyed if crossed-over with a second schema: the operator splits the valuable schema somewhere, and swaps its half strings with the second one. The one point crossover is thus not always effective, because not only does it not use the good data, but it may even destroy it. [6] also raises an additional argument against this crossover. The amount of mutated genes depends on the length m of the chromosome, $P_M \cdot m$ on the average. This allows a control of the exploration intensity. One point crossover always combines two strings together, regardless of the length.

Higher order crossover operators have been developed. A two-point crossover [38], in which the selected chromosomes are cut at two positions and then swapped overcomes the above disadvantage. For example, two relatively good schemata S_1=(110|* * * * * |0101) and S_2=(* * * |* 111 * |* * * *), where the random split locations are also displayed, procreate the new couple S'_1=(110|* 111 * |0101) and S'_2=(* * * |* * * * * |* * * *). The schema theorem claims that the chromosome S'_1 has a higher fitness than its parents. Of course, the two-point crossover destroys more sophisticated schemata, and so on. [38] has experimented with a multi-point crossover as an extension of this idea. In such a crossover a randomly even number of splits, from [0,m-1], is generated, and the segments are swapped. More advanced crossover operators are described in [6][2]. The most significant conclusion [6] reaches is that while the effectiveness of high order crossover operators may vary from case to case, it is always better than the classical crossover.

Since in this study the six degree chromosome is relatively short, only the two-point crossover operator is considered. For any individual selected for crossover (at identical probability of 0.25), two uniformly distributed discrete numbers, $r_1 \in [1,5]$ and $r_2 \in [r_1,5]$, are randomly generated (the upper limit is 5 because for six genes five split positions exist) to determine the positions of the splits. Note that if r_1=5, there is decadence to a regular crossover operation, because this enforces $r_2=r_1$. Experimental results are given in Table 4.4.

Run	1	2	3	4	5	6	7	8	9	10	Ave
F_P	0.1271	0.1268	0.1281	0.1275	0.1281	0.1278	0.1275	0.1280	0.1276	0.1278	0.1276
gen	30	54	85	67	97	48	73	81	52	47	63.4

Table 4.4. *Experimental results with two-point crossover.*

The two-point crossover offers better performance than the one-point, as is expected.

4.5 Arithmetical Crossover Operator

Similar to the effect of uniform mutation discussed in Section 4.2, the classical crossover operation might also offset the algorithm from a promising regime if the change is too significant. A possible better breeding mechanism is to procreate offspring in the

[2] Section 4.3, page 67.

vicinity of the parents, as does the non-uniform mutation operator. An arithmetical crossover is defined as a linear combination of the parents,

$$\begin{aligned} x_i(gen+1) &= a \cdot x_j(gen) + (1-a)\cdot x_i(gen) \\ x_j(gen+1) &= a \cdot x_i(gen) + (1-a)\cdot x_j(gen) \end{aligned} \tag{4.4}$$

where $x_i(gen)$ and $x_j(gen)$ are the parent chromosomes at generation *gen*, whereas $x_i(gen+1)$ and $x_j(gen+1)$ are their children at generation *gen+1*. The pre-declared parameter a determines the proximity of the offspring to the parents: the more it is decreased, the closer are the children. When a=0.5, a simple mathematical average is taken. The function can be applied to the whole chromosome vector or just to the swapped segments. In this work the second option is chosen, because the first method actually does not juxtaposition building blocks, but changes the entire string, thus contradicting the schema theorem. However, as in Section 4.1, a variable parameter a is included,

$$a = -0.9\frac{gen}{gen\_max} + 1 \tag{4.5}$$

Thus a runs linearly between 1 and 0.1 as the search advances. As the process converges, the new solutions are closer to their parents. Experimental results with this alternative are shown in table 4.5.

Run	1	2	3	4	5	6	7	8	9	10	Ave
F_P	0.1278	0.1274	0.1279	0.1275	0.1270	0.1277	0.1277	0.1278	0.1280	0.1274	0.1276
gen	63	53	100	50	30	59	78	79	96	65	67.3

Table 4.5. *Experimental results with arithmetical crossover.*

The table shows that arithmetical and two-point crossover give a similar search efficiency. The small difference in the speed is negligible.

4.6 Experimental Results with Combined Methods

The success with the application of various single evolutionary operators to the algorithm suggested experiments with multiple mixed operators. The non-uniform selection pressure, the non-uniform mutation and the advanced crossover alternatives give better results but converge slower. While Baker's reproduction does not improve the results, it may increase the speed of convergence due to its special genetic data collection mechanism. Different combinations were tested, the results are shown in Tables 4.6 and

4.7. As usual, 10 runs were executed for each configuration. In all runs non-uniform selection pressure and non-uniform mutation operator were incorporated.

Crossover	1 point		2 point		Arithmetical	
Selection	Ranked	Baker	Ranked	Baker	Ranked	Baker
1	0.1279	0.1280	0.1281	0.1280	0.1275	0.1281
2	0.1278	0.1281	0.1271	0.1281	0.1276	0.1274
3	0.1281	0.1281	0.1280	0.1280	0.1281	0.1280
4	0.1281	0.1275	0.1281	0.1281	0.1276	0.1281
5	0.1280	0.1281	0.1281	0.1281	0.1281	0.1281
6	0.1281	0.1281	0.1278	0.1279	0.1278	0.1281
7	0.1279	0.1277	0.1272	0.1274	0.1280	0.1277
8	0.1276	0.1278	0.1281	0.1277	0.1281	0.1278
9	0.1273	0.1280	0.1276	0.1275	0.1280	0.1274
10	0.1273	0.1280	0.1278	0.1280	0.1280	0.1273
Average	0.12781	0.12794	0.12779	0.12788	0.12788	0.12780

Table 4.6. *Experimental results with combined evolutionary operators.*

Crossover	1 point		2 point		Arithmetical	
Selection	Ranked	Baker	Ranked	Baker	Ranked	Baker
1	67	62	84	65	76	79
2	71	66	71	86	84	46
3	90	67	100	48	78	49
4	100	49	47	77	90	67
5	78	70	94	91	78	86
6	73	65	52	48	63	60
7	87	62	61	49	89	75
8	51	46	73	51	88	86
9	89	59	47	56	91	61
10	74	71	72	56	80	76
Average	78.0	61.7	70.1	62.7	81.7	68.5

Table 4.7. *Experimental convergence results with combined evolutionary operators.*

The results are quite uniform, without any prominent preference when only Table 4.6 is considered. All the combinations have converged faster with Baker's mechanism, as anticipated, offering reduced running time. The interesting discovery is that none of the combined algorithms have given better results than those already obtained for the non-uniform mutation operator alone, described in Section 4.2. The combination of the non-uniform selection pressure and non-uniform mutation operator, with or without high order crossover, does not improve on the performance of the algorithm with only the non-uniform mutation, Table 4.2. Table 4.8 shows the experimental results obtained when a uniform selection pressure of 0.94 and regular one-point crossover operator were combined with the non-uniform mutation operator and Baker's selection.

Run	1	2	3	4	5	6	7	8	9	10	Ave
F_P	0.1279	0.1279	0.1281	0.1280	0.1280	0.1281	0.1281	0.1281	0.1281	0.1280	0.1280
gen	50	56	40	72	68	58	70	79	60	42	59.5

Table 4.8. *Experimental results with uniform pressure, non-uniform mutation and Baker's selection.*

The results with a uniform selection pressure are better. The robustness of this configuration is remarkable, and the accuracy is more than 99.75% . The fine local tuning is apparently a vital part in an effective search. For the one-point crossover, for example, the average has significantly increased to the same value as in all other runs. Before using the advanced operators, the average for both rank-based and Baker's selection was only 0.1272, see Tables 3.2 and 4.3.

In an effort to select the best configuration for the search, additional analysis displayed in Table 4.9 has been done. For each combination of evolutionary operators, the standard deviation of the results and the number of times that the overall optimum has been reported have been computed, representing the distribution of the results.

	Configuration	F_P	*gen*	10^6-^{2}	N_{MAX}
1	Datum	0.1272	38.3	597	0
2	1 + Non uniform selection pressure	0.1275	74.3	429	2
3	1 + Non uniform mutation	0.1279	60.7	266	4
4	1 + Baker's selection	0.1272	48.7	770	2
5	1 + two point crossover	0.1276	63.4	405	2
6	1 + Arithmetical crossover	0.1276	67.3	282	0
7	1+2+3	0.1278	78.0	295	3
8	1+2+3+4	0.1279	61.7	196	4
9	1+2+3+5	0.1278	70.1	359	4
10	1+2+3+4+5	0.1279	62.7	244	3
11	1+2+3+6	0.1279	81.7	223	3
12	1+2+3+4+6	0.1278	68.5	313	4
13	1+3+4	0.1280	59.5	78.1	5

Table 4.9. *Comparison of all evolutionary strategies experimental results.*

Option 13, with uniform selection pressure and non-uniform mutation, Baker's selection and one-point crossover, is the best alternative. Among the configurations it has the highest average payload fraction, is the fastest, has the lowest standard deviation, and has found the optimum of 0.1281 more times than the others.

4.7 Conclusions

Four lessons have been learned from the runs performed so far. First, the non-uniform mutation operator is a vital tool in any genetic exploration. The contribution of other evolutionary operators in the unconstrained optimization is, surprisingly, negligible, even negative in some cases. The possible reason is the relatively short chromosome length adopted in this research, negating the influence of modified crossover methods, for example. Second, the resolution shown in Table 3.1 is not enough, because better payload fractions are achieved using intermediate values. Only the non-uniform mutation, which makes the search continuous, is able to explore these values. Thus, when using a floating point representation it is advisable to create new genes, both in the initialization and mutation, in a continuous and not a discrete manner. In this way the designer is not worried whether his required resolution is sufficient. Third, faster convergence of up to 25% can be achieved with Baker's reproduction system. The fourth conclusion is that a genetic search has to be patient to find the optimum. There is always a trade-off in the determination of *n_converge*, the number of successive generations needed for convergence call: a low value is associated with short running time, whereas a high value allows more time for the optimization. It is recommended to permit long runs because the total number of required runs as to be sure that the actual optimum has been obtained will be lower.

A typical feature of a patient search engine is the highly saturated final population. The computer has enough time to search in the entire neighborhood of the optimal chromosome. Figure 4.4 shows the final population of a certain run, under a very elitist selection of P_S=0.94, with a maximum of 0.1281. This is the same output type as was given in Figure 2.14. Duplicated chromosomes are marked; there are 21 identical chromosomes, showing that convergence is also generation-wide, not only across generations.

```
 Cp   Cx    Cdl   Cbt   CD10   Crd    Fp    Area  kW  Vs  GM-   GM+   D10  Vd

.70  .90   75.9  2.8   15.0   .75   .1217                  x           x    x
.70  .80   75.9  3.2   15.0   .00   .1228                        x     x
.70  .90   75.9  2.8   15.0   .00   .1281                  x           x    x
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.66  .90   75.9  2.8   15.0   .00   .1254                  x           x
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   10.8   .00   .1227                  x
.70  .80   75.9  2.8   15.0   .00   .1243                        x     x
.70  .90   68.5  2.8   15.0   .00   .1277                  x           x
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
.70  .90   75.9  2.8   15.0   .00   .1281    Duplicated solution - chromosome# 3
```

```
.70 .90  75.9  2.8  15.0  .00  .1281    Duplicated solution - chromosome# 3
.70 .90  75.9  3.2  15.0  .00  .1268                           x
.70 .90  75.9  2.8  15.0  .00  .1281    Duplicated solution - chromosome# 3
.70 .90  75.9  2.8  15.0  .00  .1281    Duplicated solution - chromosome# 3
.70 .90  75.9  2.8  15.0  .0U  .1281    Duplicated solution - chromosome# 3
.70 .90  75.9  2.8  15.0  .00  .1281    Duplicated solution - chromosome# 3
.70 .90  75.9  2.8  10.1  .00  .1214    x            x
.55 .90  75.9  2.8  15.0  .00  .1210             x             x
.70 .90  75.9  2.8  15.0  .00  .1281    Duplicated solution - chromosome# 3
```

Figure 4.4. *Saturated population at final generation.*

For the predefined payload fraction accuracy of up to the forth decimal digit, various maximums of 0.1281 have been found. The computer, at every run, picks the highest value using all of the digits, but the data collected through all executions is recorded with only four decimal digits Table 4.10 displays some of the chromosomes found to have the highest payload fraction.

C_P	C_X	$C_{\lambda L}$	C_{BT}	C_{DIO}	C_{RD}	F_P
0.700	0.900	77.30	2.80	15.00	0.00	0.1281
0.700	0.900	76.60	2.80	14.98	0.00	0.1281
0.700	0.900	77.03	2.82	15.00	0.00	0.1281
0.700	0.900	76.70	2.80	14.96	0.00	0.1281
0.700	0.900	76.00	2.80	15.00	0.00	0.1281
0.700	0.900	76.50	2.80	15.00	0.00	0.1281
0.694	0.900	77.46	2.80	15.00	0.00	0.1281

Table 4.10. *Optimal hulls in the unrestrained solution space.*

Except for the displacement to length ratio $C_{\lambda L}$, the genes are nearly identical, thus the hulls are geometrically similar. The identical payload fraction in Table 4.10 makes the full load weight of all unconstrained optimal ships equal. Thus their length and consequently their beam and draft differ, due to the slightly different $C_{\lambda L}$. Excluding $C_{\lambda L}$, all genes are at their extreme value, thus the maximums obtained are located at the boundary of the unconstrained solution space. At this stage the boundaries are not extended, because the results represent unfeasible ships. A similar behavior in the constrained solution space will require this operation.

Chapter 5

Optimization in the Constrained Solution Space

Until now, all efforts were made in the construction of the search/optimization engine. Since it does not matter to the algorithm what parameter is optimized, physical constraints were abandoned. Now that the algorithm is tuned, feasibility must be considered. This chapter reviews the existing techniques for constrained problems, examines their applicability, and describes the solutions applied in this research.

Although the best system parameter configuration in Chapter 4 is the one with a uniform selection pressure of 0.94, experiments in the constrained solution space, as discussed here in Sections 5.2 and especially 5.3, show that the non-uniform pressure gives better results. While the unconstrained space is big, the feasible regimes are relatively small. Thus, in order to succeed in finding the overall feasible maximum, the algorithm needs to collect genetic data from many zones, until the vital material is found. The experiments reveal that the uniform selection pressure does not allow the necessary wide exploration, and hence in some cases relatively low maximums were obtained using this method. The non-uniform pressure, on the other hand, concentrates on exploration at the beginning of the process, and is able to gather the required genetic data.

5.1 Feasible Solutions in the Unconstrained Search

In Section 2.14 it was shown that feasible individuals are rare. It was also shown, in Chapters 3 and 4, that all maximums obtained are non-feasible. This is expected, since generally unrestricted systems, like an unbalanced ship, give enhanced performance. It should not be expected to find feasible solutions at advanced generations at all, otherwise there is a problem with the process - as the exploration progresses the population must be saturated with only the best chromosomes, the non-feasible ones. An experimental illustration was displayed in Figure 4.4. Even if incidentally two feasible individuals are mated by the crossover genetic operator, the chances are that their two offspring are non-feasible. The same applies for a mutated genotype. If we do find a feasible solution, it is randomly generated, and will be destroyed soon, unless it has a relatively good performance, or has sneaked through the selection filter. Figure 5.1 demonstrates the decay of the number of feasible solutions during an unconstrained experiment. This run was randomly initialized with 5 feasible chromosomes, a relatively high number. These individuals had relatively high fitness, and they bred to some extent. After the 10th generation their appearance was incidental: only one feasible solution was obtained in the final two generations. Most of the time no feasible solutions were found in the population.

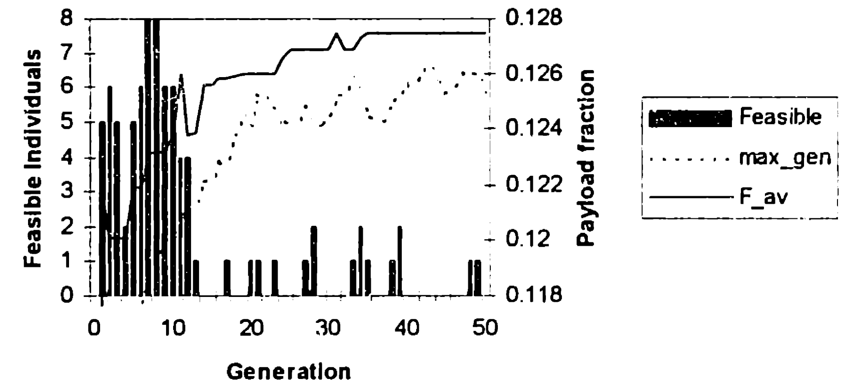


Figure 5.1. *Number of feasible individuals at unconstrained exploration.*

However, the selection theory mathematically states that the average fitness of the population increases with time. Hence, feasible genotypes must have relatively high payload fraction to be created. Therefore, the optimization must work on feasible individuals also, if they do exist in the population. Figure 5.2 displays the feasible maximum and average generational fitness, and illustrates the operation of the algorithm on feasible solutions in the unconstrained search. The process began to converge. It stopped only because of earlier non-feasible convergence.

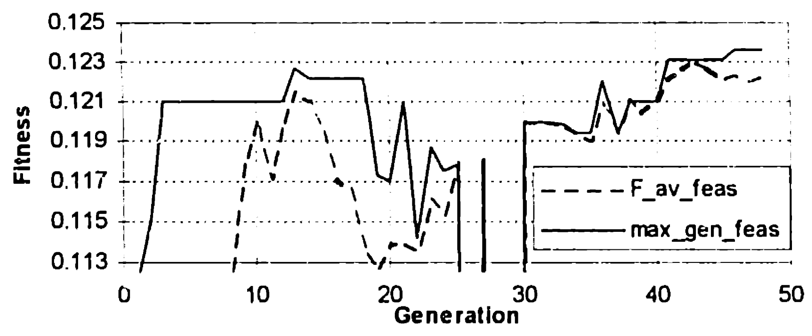


Figure 5.2. *Convergence of feasible solutions in an unconstrained exploration.*

Between generations 13 and 19 the feasible fitness has decreased, and moreover, between generations 25 and 30 it was zero (or, no feasible individuals in the population at all): the former is a result of evasion of a poor solution through the selection mechanism; the later is an example of distinction of feasible solutions, because they offer relatively low payload fraction. In cases for which the average fitness and the maximum fitness coincide, only one feasible individual existed in the population. Figure 5.2 represents the result of a population initialized with many feasible solutions, such that the algorithm had enough time to maximize them too, before they were destroyed. In the majority of runs, at the end of the process no feasible solutions survive, and poor best feasible results were obtained, in the order of 0.1200.

Motivated by the future consideration of feasible solutions, during all runs performed in Chapters 3 and 4, the best feasible solutions were also recorded. Table 5.1 shows the best recorded results. It represents the results of a somewhat random search, because in those runs the model was programmed to maximize the overall payload fraction. As illustrated in Figure 5.1, the appearance of feasible chromosomes at late stages is accidental. Nevertheless, it still provides good information and some idea of the expected results when exploring the constrained solution space. The final results in the research must be higher or at least equal to those in Table 5.1.

C_P	C_X	C_{ML}	C_{BT}	C_{DIO}	C_{RD}	F_P
0.70	0.90	77.4	3.2	13.9	0.00	0.1257
0.70	0.90	67.4	3.1	14.5	0.00	0.1256
0.70	0.90	81.4	3.1	13.7	0.04	0.1255
0.70	0.88	68.0	3.1	14.5	0.00	0.1254

Table 5.1. *Best feasible results obtained in the unconstrained search.*

As in Table 4.8, the proximity of the genes, except for C_{ML}, shows that the direction of the search is right. All hulls have similar shape, the only difference is the exact dimensions for length, beam and draft.

5.2 Optimization with the Racist Model

Clearly, the feasible chromosome is not stable. The feasible regime is so small, that feasible ships in the unconstrained environment experience extinction: typically even offspring of two feasible parents are non-feasible. In nature, to prevent extinction, species are isolated and transferred to a more comfortable environment, where they can breed without danger. The *racist model* was built according to this analogy. The algorithm is described in Figure 5.3. Being based on the code of Chapter 4, the list file of this model is not included.

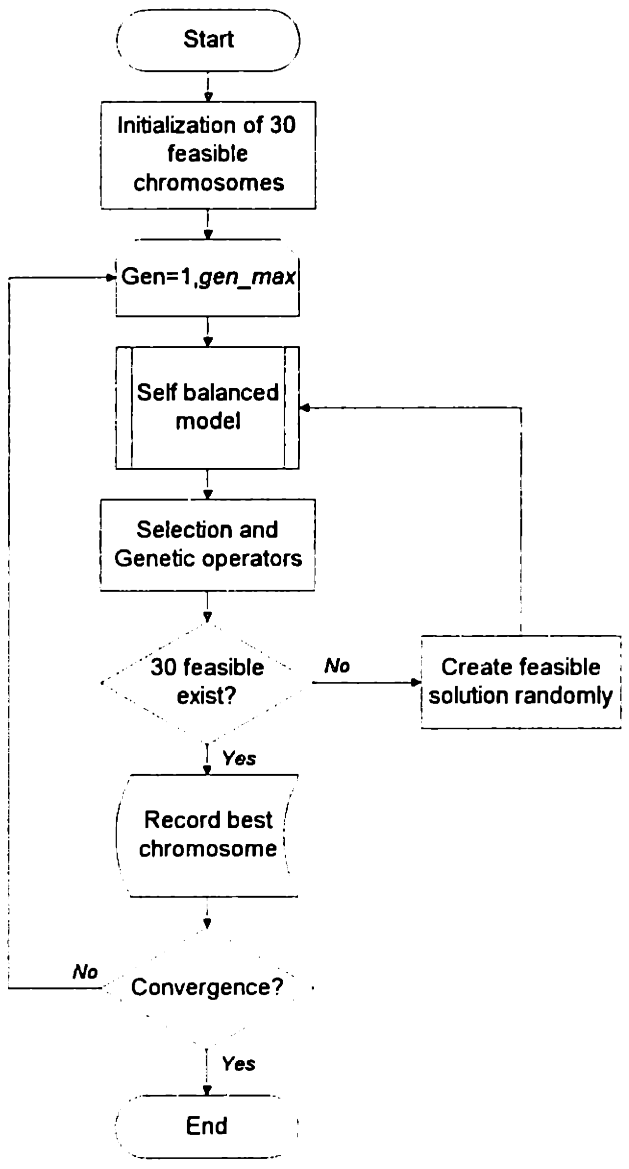

Start
Initialization of 30 feasible chromosomes
Gen=1,gen_max
Self balanced model
Selection and Genetic operators
30 feasible exist?
No
Create feasible solution randomly
Yes
Record best chromosome
No
Convergence?
Yes
End

Figure 5.3. *Racist optimization model flowchart.*

Only feasible solutions are allowed - "dangerous" non-feasible chromosomes are not wanted and destroyed. The population is initialized randomly with only feasible members, until *pop_size* of them exist. The nearness of the respective genes in Table 5.1 implies that the payload fraction function is continuous. Thus, a feasible new solution would be generated if the original gene change is small. The two genetic operators which fulfill exploration in the vicinity of former individuals are the arithmetical crossover (discussed in Section 4.5) and the non-uniform mutation (Section 4.2). For the former, a constant value of a=0.05 is set, and for the later b=5 is utilized. If a non-feasible chromosome is obtained, it is deleted and the computer generates randomly a new feasible individual. The optimizer uses the other features of the search engine as concluded in Chapter 4: non-uniform selection pressure[1] and Baker's roulette wheel.

Undoubtedly, this model has many drawbacks. First and most significant, it contradicts the selection theory because new species are thrown into the population without being evolved in time. The whole progress achieved by the evolution process is contaminated with primitive individuals, which return the average fitness back in time. If for example relatively fit parents produce non-feasible children, the computer destroys them and creates two random feasible chromosomes, likely to be less fit because of the random generation. An experimental demonstration is shown in Figure 5.4. The figure displays the maximum and average payload fraction at each generation, and the number of times the computer created a new feasible solution. On average for this run, about 4 chromosomes are thrown into the population each generation. While the maximum value improves with time, the average oscillates due to this addition and does not improve with time.

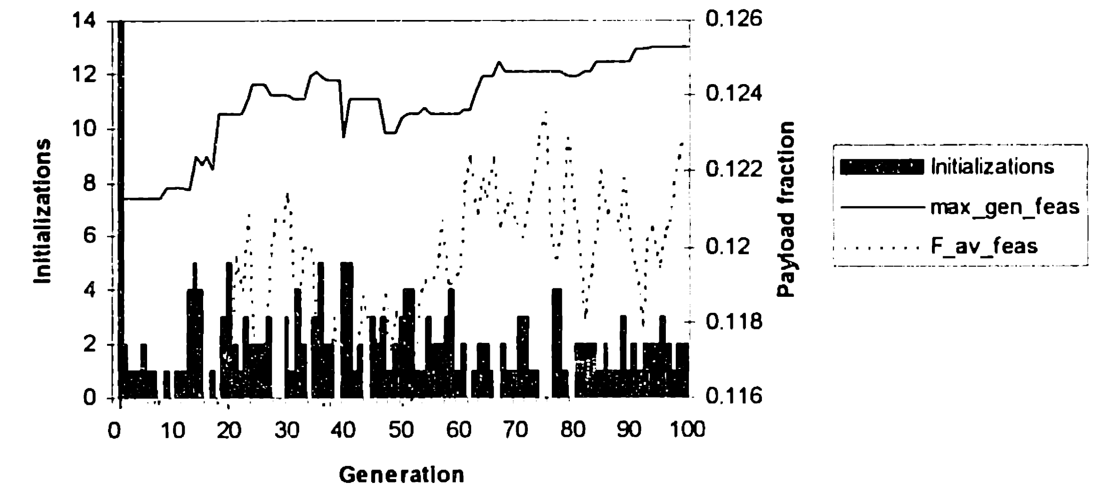


Figure 5.4. *Contradiction of the evolution principle in the racist model.*

Secondly, the gentle genetic operations increase the chances of feasibility, but ignore the possible existence of better solutions in other regimes, which are not accessed because none of the initialized individuals are located there. An increased population size would increase the number of visited feasible areas, thus reducing this disadvantage, but it takes a lot of time to randomly create or complete *pop_size* feasible chromosomes, because they are so rare. Creation of 10 feasible genotypes takes approximately 20 minutes, effectively more than ten times longer than usual, and a complete run is not practical. The long running time is a great disadvantage. Fourth, the discarded non-feasible solution might

[1] See in the introduction of this chapter.

contribute some necessary genetic data, and even create a feasible solution, as illustiated in Figure 5.1.

Experimental runs show that the method does work. Table 5.2 provides the results of 5 executions. Due to the long running time, populations of 10 and 15 members were used. The number of trials for which the computer tries to create a substitute feasible solution is bounded to 15. When this limit is reached the program randomly takes an existing feasible individual from the same generation as the replacement. The number of identical successive generations needed for convergence is reduced to 10.

	10 individuals		15 individuals	
	F_P	*gen*	F_P	*gen*
1	0.1242	100	0.1248	100
2	0.1241	100	0.1253	100
3	0.1250	100	0.1255	100
4	0.1237	100	0.1255	100
5	0.1244	100	0.1247	100
Ave	0.1243	100.0	0.1252	100.0

Table 5.2. *Feasible experimental results with the racist optimization model.*

The effect of the population size is easily seen. Not only is the average fitness higher, but also the distribution of the results is more uniform. Should the model be run with a bigger population, the results would be even better. This was not done for running time reasons. The fact that none of the runs has converged implies a high mutation rate. It was decided not to decrease this rate because it proved itself in Chapter 4. The racist optimizer gives poor results in reference to those of Table 5.1, but while Table 5.1 represents the best performance found in approximately 350 runs, most of them yielded low fitness, the distribution here is uniform. The current model is more robust, even for small populations. The premature convergence is the result of the selection theory violation, and the lack of required genetic material.

5.3 Optimization with the Penalty Function

The approach of Section 5.1 is not useful, since it requires a large number of executions in the unconstrained space, which makes it not effective, and the results uncertain. The racist model does use the features of the effective algorithm constructed in Chapters 3 and 4, but the addition of new chromosomes into the population disturbs the process and causes premature convergence. The solution should combine the features of these approaches: use the data of non-feasible chromosomes, but concentrate on the feasible members. This requires the computer to distinguish quantitatively between them, in a systematic way.

Linear systems with constraints are easy to optimize in a systematic way. Michalewicz [6] created the *GENOCOP* system to deal with such cases. The model converts the set of equations and inequalities to expressions limiting each gene value as a function of themselves. Then, when a gene is selected for update, by either the crossover or mutation operator, it can get a value only from this range, and the new chromosome remains feasible. For non-linear problems the method is not applicable, particularly for problems without an explicit mathematical representation. A second method is to use *recovery algorithms*: when a non-feasible chromosome is found, the computer randomly chooses a gene, and then goes back and somehow finds the domain of that gene for which the chromosome is recovered and returns to be feasible. The value of that gene is randomly chosen from that range. Such methods are not practical in large systems such as the DDGx model: the running time is too long even on a fast computer. Most of the references recommend using a *penalty function*. In this method, the objective function incorporates knowledge about the feasibility of the solution. Fitness of a non-feasible individual is reduced (or increased in minimization problems). The reduction is by means of a penalty,

$$F_p^{Effective} = F_p - Penalty \tag{5.1}$$

The penalty is associated with the distance from which a gene has deviated from its allowed domain. The bigger the constraint violation, the bigger the penalty. In problems with more than one constraint, the penalties are summed,

$$F_p^{Effective} = F_p - \sum_i Penalty_i \tag{5.2}$$

The difficulty is to decide on the intensity of the penalty. Exaggerated punishment suppresses non-feasible solutions and their vital genetic information. A weak penalty, on the contrary, does not focus on feasible solutions. At this point one of the biggest advantages of the rank-based selection mechanism solves the dilemma. Revisiting Section 3.4, in this method probabilities of survival are assigned to the chromosomes according to their relative rank when fitness values are sorted in descending order and not according to their relative fitness: the best one is positioned as number one, the second best gets the second rank, and so on. There is no importance to the relation between the values, their proximity for example. From a different point of view, this is a serious drawback, because very poor individuals can also survive. However, in this proven evolutionary program it is irrelevant. Therefore, no matter how strong the penalty is, provided that it is applied evenly to all of the non-feasible solutions, their position in the sorted population does not change, and they still get the same chance to survive. Their necessary genetic data is not lost, and the algorithm can use it for the optimization. Similarly, a tough penalty policy might turn some of the values to negative. In a regular roulette wheel the associated probability would be negative, which has no meaning; in a rank-constructed wheel this cannot occur.

The penalty function used is based on the relative deviation of each gene. Six constraints are defined in this optimization problem, one of which - the initial stability - restricts the respective value from two directions,

$$C_{GMB}^{Min} \le C_{GMB} \le C_{GMB}^{Max} \tag{2.105}$$

The rest limit the values from one side of the plane only,

$$V_S \ge 29.96 \quad knots \tag{2.90}$$

$$D_{10} \ge D_{10\min} \tag{2.94}$$

$$KW_G \ge KW_{GREQ} \tag{2.96}$$

$$A_{DA} \ge A_{DR} \tag{2.98}$$

$$V_D \le V_{D\max} \tag{2.101}$$

The electrical load constraint has never been violated during the runs. For that reason this constraint is disregarded for the penalty. Area, speed, stability, depth and deckhouse volume errors are defined,

$$\begin{aligned}
ERR_{Area} &= \frac{A_{DR} - A_{DA}}{A_{DR}} \cdot 100 \\
ERR_{Speed} &= \frac{V_S^{Min} - V_S}{V_S^{Min}} \cdot 100 \\
ERR_{Stabil} &= \frac{C_{GMB}^{Min} - C_{GMB}}{C_{GMB}^{Min}} \cdot 100 \quad or \quad \frac{C_{GMB} - C_{GMB}^{Max}}{C_{GMB}^{Max}} \cdot 100 \\
ERR_{D10} &= \frac{D_{10\min} - D_{10}}{D_{10\min}} \cdot 100 \\
ERR_{VD} &= \frac{V_D - V_{D\max}}{V_{D\max}} \cdot 100
\end{aligned} \tag{5.3}$$

The errors are expressed in percentages deviation, and are always positive. The net penalty is determined as reduction of 0.0001 from the real payload fraction per each k percent total deviation, and Equation (5.2) becomes

$$F_P^{Effective} = F_P - 0.0001 \cdot \frac{\sum_i ERR_i}{k} \tag{5.4}$$

Negative errors are considered as zero in this summation, namely, the gene does not get a prize on being within its allowed limits. [6] incorporates time dependence, in which the penalty is increased as the process advances, but this is not employed here. Taking advantage of the above described property of a rank-based selection, the parameter *k* is experimentally increased until the algorithm converges to a feasible individual. A first guess for *k* is obtained form the known unconstrained optimum of 0.1281 and constrained optimum of 0.1257. In order to bring the fitness from 0.1281, with net error (see Figure 3.13) of 34.6%, a reduction of 0.0024 is required, thus k= 34.6/24≈1.45. Experiments reveal that this value is not sufficient, because other non-feasible solutions are found, although with fitness lower than 0.1281. Several experiments with decreased *k* were performed, until k=0.1 was found to be satisfactory. Table 5.3 gives the experimental results of 10 different executions with this parameter. All the chromosomes represent feasible ships. The search process is identical to the configuration selected in Chapter 4, with non-uniform mutation with b=2, Baker's reproduction and classical crossover, excluding three modifications: it uses non-uniform selection pressure as explained at the beginning of the chapter, penalty to non-feasible individuals is applied, and, as concluded in Chapter 4, the search is changed to continuous. Thus, initialization is performed not according to Equation (3.2), but by

$$C = C_{\min} + (C_{\max} - C_{\min}) \cdot RAN\# \tag{5.5}$$

where now *RAN#* is a uniformly distributed random number from [0,1]. The population size is again 30. The listing of this code is included in Appendix F.

	C_P	C_X	C_{ML}	C_{BT}	C_{DI0}	C_{RD}	GM/B	F_P	*gen*
1	0.700	0.881	71.58	3.03	14.33	0.00	0.0900	0.1257	71
2	0.700	0.899	65.47	3.19	14.66	0.01	0.0910	0.1253	61
3	0.700	0.900	68.95	3.22	14.50	0.00	0.0967	0.1258	80
4	0.700	0.900	75.77	3.13	14.04	0.00	0.0909	0.1260	74
5	0.700	0.866	86.00	2.86	13.50	0.00	0.0910	0.1252	95
6	0.700	0.900	89.75	3.05	13.29	0.00	0.0911	0.1258	94
7	0.697	0.898	90.00	3.03	13.26	0.00	0.0907	0.1257	72
8	0.700	0.898	90.00	3.08	13.00	0.00	0.0923	0.1253	65
9	0.700	0.900	70.40	3.18	14.36	0.00	0.0922	0.1259	75
10	0.698	0.900	81.78	3.08	13.67	0.00	0.0900	0.1258	66
Ave								0.1257	75.3

Table 5.3. *Feasible experimental results with penalty function.*

As usual *gen* represents the number of generations required for convergence. The results are not much better than those obtained in Tables 5.1 or 5.2, but they are better for all runs. The search is also more robust, with a standard deviation of $10^6 \cdot \sigma^2$=265. Figures 5.5 and 5.6 show experimental results of the ninth run. The first displays the maximum and average fitness as a function of the time, and the second the corresponding number of feasible members in every generation.

The number of feasible solutions was expected to grow until they saturate the population with *pop_size* feasible individuals. In Figure 5.6, however, the amount of feasible chromosomes fluctuates in a random fashion, and is never close to *pop_size*; it has even decayed several times.

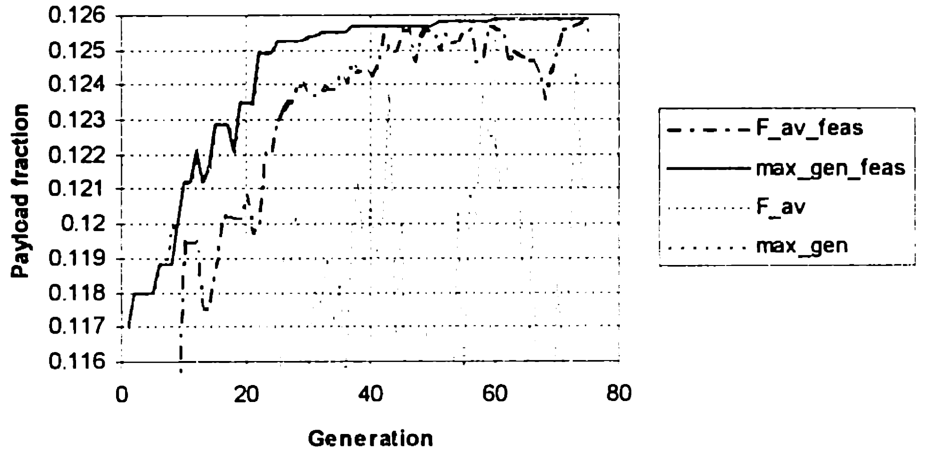


Figure 5.5. *Experimental maximum and Average fitness with punishment.*

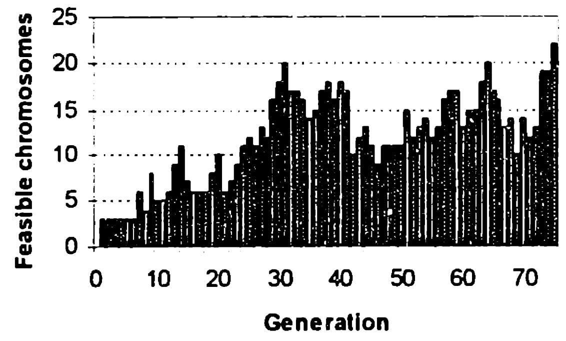


Figure 5.6. *Number of feasible chromosomes per generation for Figure 5.5 run.*

The unpredicted behavior is explained by the fact that all ships in Table 5.3 have marginal stability factor, close to C_{GMB}=0.09. The non-uniform mutation operator explores more and more in the vicinity of the best chromosomes. The algorithm walks on the stability border line in an effort to optimize the fitness. Sometimes it falls inside the allowed domain, and other times it does not. Since the decision of non-feasibility is sharp, and a new genotype is considered non-feasible even with C_{GMB} as close as 0.089999, many chromosomes in the population are not feasible. The effect of these is represented in the low overall average fitness, which is far lower than the feasible average. Had the optimal value had its constrained parameters far from the bounding limits, this would have not happened. As an example, the characteristics of the optimal hull are given in Figure 5.7. The *GM* to *B* ratio for this ship is 0.0909.

```
Main Parameters ...

  Cp                                        .700
  Cx                                        .900
  Cdl                                       75.77 [lton/ft^3]
  Cbt                                      3.130
  CD10                                     14.04
  Crd                                       .000

  Main Dimensions on Waterline ...

  Underwater volume                         2.562E+05 [ft^3]
  Length on waterline                      458.95 [ft]
  Beam                                      52.67 [ft]
  Draft                                     16.83 [ft]
  D10                                       32.69 [ft]

  Speed and Resistance ...

  Wetted area                              29283.9 [ft^2]
  Sustained speed                          30.31 [knt]
  Maximal speed                            32.23 [knt]
  Endurance shaft horsepower               16618.3 [hp]

  Space Available ...

  D0                                       45.27 [ft]
  D10                                      32.69 [ft]
  D20                                      34.17 [ft]
  Total hull volume                         671759.0 [ft^3]

  Electrical Loads ...

  Winter cruise electrical load            3308.4 [kW]
  Marginal winter cruise electrical load   3341.5 [kW]
  24 hours average electrical load         2223.1 [kW]
  Power required per generator             1856.4 [kW]
  Auxiliary machinery rooms volume         55188.8 [ft^3]

  Tankage ...

  Fuel weight                              1218.9 [lton]
  Fuel tanks total volume                  55221.1 [ft^3]
  Ballast tank volume                      10492.0 [ft^3]
  Total tanks volume                       73169.5 [ft^3]

  Weight and Center of gravity ...

  Lightweight                               5578.2 [lton]
  Vertical CG of lightship                 21.46 [ft]
  Full load displacement                    7288.8 [lton]
  Vertical CG at full load                 19.97 [ft]
  Payload fraction                          .126

  Area/volume Balance ...

  Required hull area                       50352.2 [ft^2]
```

```
Available hull area                       37971.9 [ft^2]
Required hull volume                      536754.0 [ft^3]
Available hull volume                     404780.7 [ft^3]
Required deckhouse area                   10091.9 [ft^2]
Available deckhouse area                  22472.1 [ft^2]
Required deckhouse volume                 107579.5 [ft^3]
Available deckhouse volume                239553.0 [ft^3]
Total required area                       60444.0 [ft^2]
Total available area                      60444.1 [ft^2]
Total required volume                     644333.5 [ft^3]
Total available volume                    644333.7 [ft^3]

Initial Stability .....

KB                                         9.37 [ft]
BM                                        15.39 [ft]
Metacentric height                         4.79 [ft]
GM to B ratio                              .091

Deckhouse volume                          239553.0 [ft^3]

Balance/Feasibility Status ...

Category                              Required/Minimal    Available
Deckhouse area [ft^3]                     10091.9           22472.1
Electric plant [kW]                       1856.4            2500.0
Sustained speed                           29.96             30.31
Initial stability                          .090             .0909
                                           .135
Depth [ft]                                32.66             32.69
Maximal deckhouse volume [ft^3]           315833.9          239553.0
Payload fraction                           .075             .1260
```

Figure 5.7. *Self balanced DDGx optimal ship characteristics.*

Figure 5.8 illustrates the marginal stability of explored ships at late stages. It shows the final population of the run of Figures 5.5 and 5.6, converged to the value of 0.1259 after 75 generations. The population is saturated with the best chromosome as was demonstrated already in Chapter 4 for the unconstrained exploration. Almost all non-feasible members across the population are for low stability reasons.

```
 Generation #  75

 Cp  Cx   Cdl  Cbt  CD10  Crd     Fp  Area  kW  Vs  GM-  GM+  D10  Vd

.70 .90  70.4  3.4  14.4  .00  .1252
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259   Duplicated solution - chromosome# 3
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.6  13.1  .00  .1229
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  11.3  .00  .0530    x            x
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.4  14.4  .00  .1252
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259
```

```
.70 .90  70.4  3.6  13.0  .00  .1227
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259    Duplicated solution - chromosome# 16
.70 .90  70.4  2.9  14.4  .00  .0950                 x
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .71  70.4  3.2  14.4  .00  .0372                      x
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259    Duplicated solution - chromosome# 22
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.6  .00  .1247                           x
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  61.7  3.2  14.4  .00  .1169                 x
.70 .90  70.4  3.2  14.4  .00  .1259
.70 .90  70.4  3.2  14.4  .00  .1259
```

Figure 5.8. *Final population with penalty function for the run of Figure 5.5.*

In Section 2.14, which gave experimental results after one generation only, it was recognized that the majority of stability failures were for excessive stability. It was annotated that this behavior is not typical of ship design, in which the problem is usually poor stability. The constrained optimization results in this chapter confirm that prediction.

The proximity of D_{10} to D_{10min}, as can be seen in Figure 5.7, for example, is not surprising either, because minimum weight means minimum depth. In some runs the values have coincided.

5.4 Final Local Optimization

The algorithm of the last section found the optimal value only one time. Holland [5] himself suggested to conduct a final local search, in the neighborhood of the optimal chromosome, in order to further reduce the possibility of premature convergence. His rationale was that the evolutionary search only gives the direction and tries to improve the performance of the entire population; it does not optimize a specific individual. He recommended using the genetic search as a preprocessor, to identify the high performance region in the solution space, and then to "invoke a local search routine to optimize the members of the final population". The hill climbing technique [10] is employed for the local optimization performed. The best chromosome found by the genetic search is taken as the start (or datum) point of the local search. Then, each randomly selected gene is changed by plus or minus its required precision, and the new chromosome is evaluated. Since it was discovered that the initial precision of Table 3.1 is not discrete enough, and that there is a big sensitivity to the values, the steps by which the hill climbing tries to advance are set to the values shown in Table 5.4.

Gene	C_P	C_X	$C_{\Delta L}$	C_{BT}	C_{D10}	C_{RD}
Δ	0.001	0.001	0.01	0.01	0.01	0.01

Table 5.4. *Gene steps in hill climbing optimization.*

If the new fitness is higher, the datum point gets the genes of this better chromosome and the process is repeated similarly. The source file is provided in Appendix G. Non-feasible improved results are discarded and considered as no improvement. The limit is set to 50 allowable iterations after an improvement has been reached. The algorithm did not find solutions better than the optimum of 0.1260, thus suggesting that this is an overall maxima, but worked well for other inferior runs. Table 5.4 provides the experimental results of 5 executions for a start point identical to the eighth result in Table 5.3, 0.1253.

	# of improvements	F_P
1	34	0.12580
2	11	0.12549
3	29	0.12573
4	34	0.12580
5	34	0.12580
Ave	28.4	0.12572

Table 5.5. *Hill climbing experimental results, start point at 0.1253.*

The central column gives the number of times the algorithm found a better chromosome (or, number of steps the algorithm climbed up the hill). As has already been stated in the introduction section, this technique excels in its exploitation ability. It does not care about the whole picture and about other regimes of the solution space. The results thus depend strongly on the start point. However, after the space has been investigated by the evolutionary program, the start point is the best possible guess, thus the chance of obtaining the overall optimum is very high.

5.5 Conclusions

As was known from previous chapters and expected, the fitness of feasible solutions is lower than non-feasible ones. Even from the philosophical aspect, the latter have superior performance as they are simply not constrained by laws. The weight optimized feasible ship weighs only 7289 ltons at full load, more than 10% less than the DDG51 actual design.

Among the three approaches used to optimize the constrained solution space, only the last one, the penalty function, is practical due to the better results it provides and also due to the reasonable running time and computational effort needed. All three give close results. The motivation to first construct the search engine in the free solution space, and to establish its reliability, was good. A well built robust evolutionary model functions in any environment, including constraints. It is shown that the optimization in the constrained solution space performed in this chapter is identical to the process of early chapters, with the same features: comparable convergence speed and population saturation. The collection of feasible results during the construction has given a first, evidently very good guess about the final value of the optimal payload fraction.

As the naval architect expects, and different from the experimental results of Chapter 2, the major reason for non-feasibility throughout the constrained exploration is poor stability. All reported best results in Table 5.3 have marginal low *GM* to *B* ratio. Many solutions have been punished due to this failure, and contaminated the population with non-feasible solutions, as is illustrated in Figure 5.6. This trend of attraction to low *GM/B* ratios has already started in the unconstrained optimization: the overall maximum fitness of 0.1281 has an initial stability ratio of 0.066, as shown in Figure 3.13. The final population in an unconstrained exploration, Figure 4.4 for example, has all failures associated with poor stability, except for one individual, which did fail due to excessive stability but had a significantly lower fitness.

Although no further improvement has been attained by employing the final local optimization to the optimal result of the evolutionary algorithm, the method still worked for the rest of the results, and should be incorporated in the search process.

Finally, the genes of Table 5.3 have C_P, C_X and C_{RD} at their extreme limits. The uniform values verify that the algorithm has converged in the right direction. While neither C_P nor C_{RD} associated limits can be extended, the first due to a consequent exit from C_P range in Gertler's reanalysis for the residuary drag, and the second since it must be non-negative, the C_X upper limit can be raised. The accepted upper limit to this coefficient is around 0.85. However, the physical bound is 1.0, and that was the thought at the beginning of the research when the upper value was set to 0.9. When this limit is set to 0.95, for example, payload fractions as high as 0.1268 (full load displacement of 7277.8 lton) are found. The convergence to the upper lim[illegible] C_P and C_X happens because the algorithm tries to balance the area while keeping the hull small. As the hull becomes more full, the net arrangable area increases. The conclusion is that as the maximum sectional and prismatic coefficients increase, the weight decreases. This discovery requires additional study, since usually these values for C_P and C_X are not investigated because the vessel is considered to be too "full". Typically such high values are associated with high resistance and thus are disregarded. In Chapter 6 it is shown that the residuary drag algorithm used by the DDGx model gives lower values than predicted by other methods, and thus the optimal ship here may violate the sustained speed constraint.

Chapter 6

ASSET Model of the Optimal Ship

Throughout the research the limitations and simplicity of the design model have been kept in mind. While it is proven that genetic optimization works, the pretension to compare the results to the DDG51 class is not obvious, because the prediction ability of the self-balanced model is not validated. In the preface to Chapter 2 the restrictions of the model are recognized, and it is stated that in any case the model is good only for similar hulls, with comparable characteristics. The evaluation model is calibrated at a specific point in the solution space, that of DDG51. The distance from this location, for which the results are still reasonably accurate is not known. As the distance increases the errors increase as well. In order to validate the results, the optimal hull has been modeled in ASSET, the computer tool used by the U.S. Navy for concept design. The reduction of 10% in the full load displacement is big enough that it is important to conduct this comparison. This chapter deals with this validation, and provides a comparison between the results predicted by DDGx self-balanced model and ASSET.

6.1 Model Construction

ASSET stores the vessel's description in arrays. The data bank includes a thorough description of the analyzed ship: hull, compartments, tanks, deckhouse, materials, midship section structure, power plant configuration and components, appendage and propeller. The program uses this information to compute weights, areas and volumes, and to asses the performance of the ship. The data bank created for the DDG51 match run was used as the basis for the new hull. No changes were made in the data bank to fields other than those associated with the hull or propeller determination. In this way only the effects of the different hull shape are computed. There is no intention to try to improve other aspects, such as power plant modified components, different propulsion configuration etc., because the calibration assumes that excluding hull characteristics, all other systems are identical in type but scaled in size. The parameters that were changed are given in Table 6.1. These were taken from the Fortran model output, as shown in Figure 5.7. The values of the last two parameters, which are calculated but not reported by the Fortran code, are extracted from the Mathcad model.

Parameter	Value
Prismatic coefficient C_P	0.700
Maximum section coefficients C_X	0.900
Raised deck coefficient C_{RD}	0.0
Length on design water line L_{WL} [*ft*]	458.95
Beam on design water line B [*ft*]	52.67
Draft at design water line T [*ft*]	16.83
Depth at midship D_{10} [*ft*]	32.69
Depth at station o D_0 [*ft*]	45.27
Depth at station 20 D_{20} [*ft*]	34.17
Deckhouse volume V_D [ft^3]	239553
Auxiliary machinery rooms volume V_{AUX} [ft^3]	54912
Propeller diameter D_P [*ft*]	16.69

Table 6.1. *DDGx data bank changed parameters.*

The offsets of the hull are calculated by the hull geometry module using the first nine parameters in Table 6.1. The module creates the offsets using user-defined hull boundary conditions. In order to enforce the specified D_0 and D_{20}, the hull boundary conditions indicator is set to "given" instead of "DDG51"; otherwise the computer builds the sheer line similar to DDG51, and the numerical values for these heights are different. The conceptual arrangement of the compartments is not changed, and is illustrated in Figure 6.1.

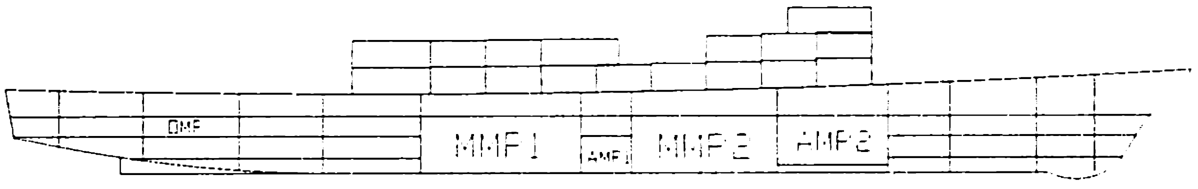


Figure 6.1. *Optimal DDGx hull general arrangement.*

The hull has one internal continuous deck and two lower decks. The height and sheer ratio of the internal decks, as well as the position of the transverse bulkheads, were changed in an iterative manner, to get the same main machinery room total volume of 138600 ft^3, auxiliary machinery room total volume of 54912 ft^3, and an average hull deck height of 10.66 ft, as in the self-balanced model. All three are calculated by the hull subdivision module, the latter by the relation

$$H_{DK}^{Hull} = \frac{V_{HA}}{A_{HA}} \tag{6.1}$$

where the notation of Chapter 2. V_{HA} is the net arrangable available hull volume, defined in equation (2.80), and A_{HA} is the net available hull area. Note that in the DDGx math model H_{DK} is a predefined parameter. The volume of machinery spaces is set as specified by the DDGx model, and no attempts were made to minimize these spaces in order to gain additional arrangable space. It is noticed that these rooms may be bigger than required.

Superstructure is defined in ASSET by means of prismoids, altogether creating the required shape of it. 17 prismoids are created in the model. As a first guess, a deckhouse as in Figure 2.12 was built. Then some prismoids were eliminated, and others reshaped iteratively, until the required volume of 239553 ft^3 is attained. An isometric view of the hull and deckhouse, obtained by exporting the data to Autocad, is displayed in Figure 6.2. As specified in Chapter 2, the superstructure has two decks, except for the bridge, which forms a third deck in the front part of the deckhouse.

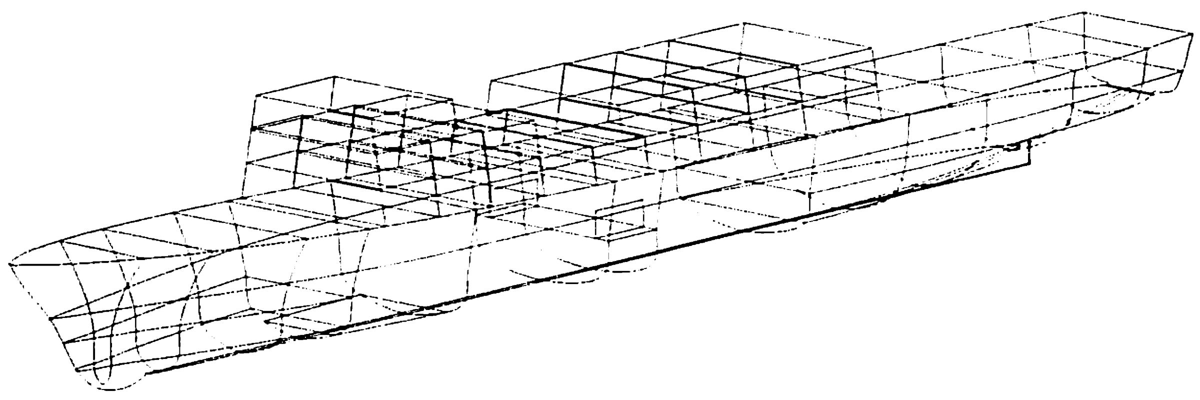


Figure 6.2. *Optimal vessel hull and deckhouse.*

The prismoids of the deckhouse are adjusted for the location of the intakes and exhausts of the propulsion plant. The mechanical arrangement, which is identical to that of DDG51, is illustrated in Figure 6.3. Each machinery room contains two LM2500 gas turbines, connected to a single gear box and driving one shaft. The rated power of each gas turbine is set automatically by the computer to 25775 hp. It is a little lower than the current specified power, but is kept to be comparable to DDG51 results, which has this rating in the match run. The aft machinery room also contains one Allison 501-K34 ship service generator. The other two generators are located at the forward auxiliary machinery room and the other machinery room. The stacks, which are not shown in the picture, are located on the superstructure. The vertical position of the main engines is adjusted in order that the two fit in the rooms.

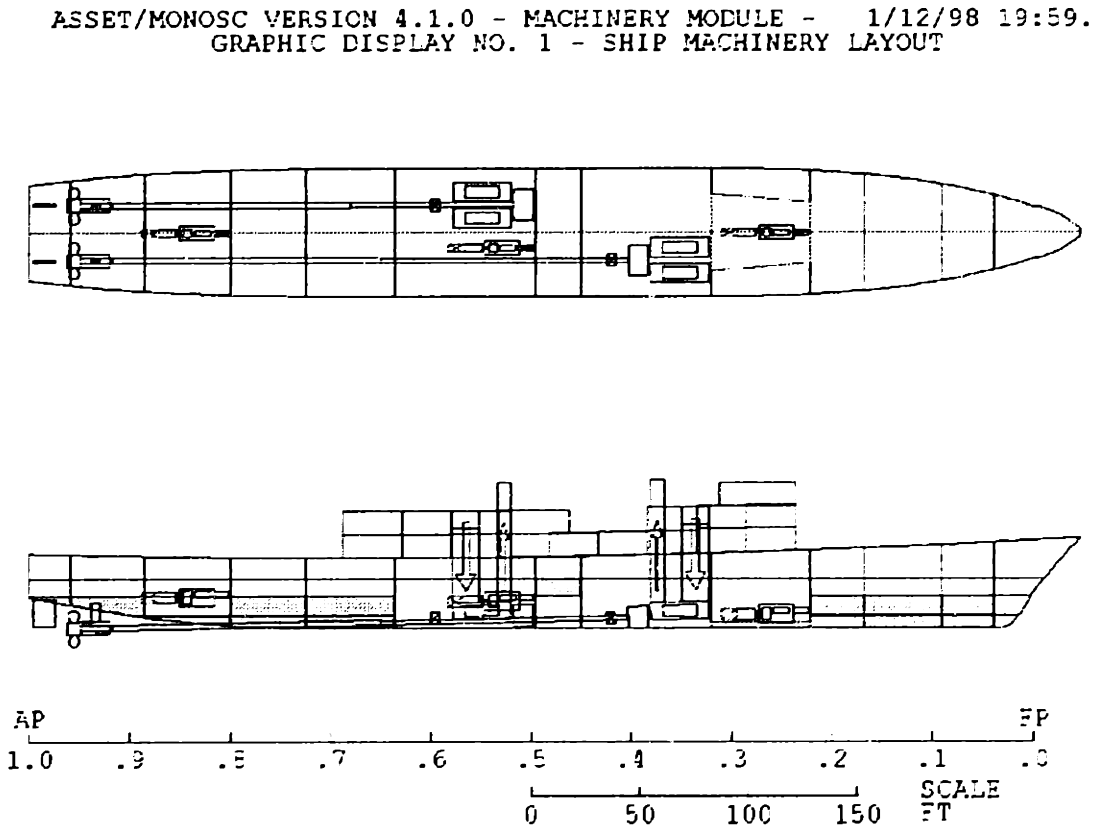

ASSET/MONOSC VERSION 4.1.0 - MACHINERY MODULE - 1/12/98 19:59.38
GRAPHIC DISPLAY NO. 1 - SHIP MACHINERY LAYOUT
AP
FP
1.0
.9
.8
.7
.6
.5
.4
.3
.2
.1
.0
0
50
100
150
SCALE
FT

Figure 6.3. *DDGx mechanical arrangement.*

The location of the fuel tanks is also shown in Figure 6.3. These are defined as tanks in the large objects array. The center tank is the service wing tank, and can also be seen in Figure 6.2. Their size was adjusted iteratively such that the exact required fuel volume, calculated by the machinery module, is allocated[1].

Last, the propeller diameter is specified as calculated by the Fortran model. All other propeller required parameters are evaluated by the propeller module. As mentioned

[1] This is a necessary step in ASSET. Otherwise, the computer assigns internal volumes as tanks automatically, until the required volume is obtained. The total assigned volume might be bigger than required, and thus arrangable area is wasted.

before, all other parameters were kept at their original value for DDG51, including design margins. The summary output of the design summary module is shown in Figure 6.4.

```
ASSET/MONOSC VERSION 4.0.0 - DESIGN SUMMARY - 11/20/97 10:50.22

PRINTED REPORT NO. 1 - SUMMARY

SHIP COMMENT TABLE
     DDGX HULL MODEL.

  PRINCIPAL CHARACTERISTICS - FT              WEIGHT SUMMARY - LTON
LBP                           459.0     GROUP 1 - HULL STRUCTURE    2482.7
LOA                           480.5     GROUP 2 - PROP PLANT         789.8
BEAM, DWL                      52.7     GROUP 3 - ELECT PLANT        306.1
BEAM, WEATHER DECK             58.3     GROUP 4 - COMM + SURVEIL     408.7
DEPTH @ STA 0                  32.7     GROUP 5 - AUX SYSTEMS        734.3
DRAFT TO KEEL DWL              16.8     GROUP 6 - OUTFIT + FURN      619.0
DRAFT TO KEEL LWL              16.4     GROUP 7 - ARMAMENT           314.9
FREEBOARD @ STA 3              24.7     ---------------------------------
GMT                             4.8     SUM GROUPS 1-7              5655.4
CP                             .700     DESIGN MARGIN                 31.2
CX                             .900     ---------------------------------
                                        LIGHTSHIP WEIGHT            5686.6
SPEED(KT):  MAX= 32.2  SUST= 30.2       LOADS                       1699.3
ENDURANCE:  3807.6 NM AT 20.0 KTS       ---------------------------------
                                        FULL LOAD DISPLACEMENT      7385.9
TRANSMISSION TYPE:             MECH     FULL LOAD KG:  FT             19.3
MAIN ENG: 4 GT         @ 25775.0 HP
                                        MILITARY PAYLOAD WT - LTON 1089.6
SHAFT POWER/SHAFT:       50270.8 HP     USABLE FUEL WT - LTON      1210.1
PROPELLERS: 2 - CP -   16.7 FT DIA

SEP GEN: 3 GT          @ 2500.0 KW
                                                OFF   CPO   ENL   TOTAL
24-HR LOAD                   2145.4     MANNING  26    24   291     341
MAX MARG ELECT LOAD          3281.5     ACCOM    26    24   291     341

      AREA SUMMARY - FT2                     VOLUME SUMMARY - FT3
HULL AREA               -    33329.     HULL VOLUME              -  632417.
SUPERSTRUCTURE AREA     -    21213.     SUPERSTRUCTURE VOLUME -     239612.
--------------------------------        ---------------------------------
TOTAL AREA              -    54542.     TOTAL VOLUME             -  872029.
```

Figure 6.4. *Design summary output for DDGx optimal ship.*

Two warning messages were received from the computer when synthesizing the data. The first one was from the propeller model, which has found that the required expanded area ratio is 1.09, while the accepted maximal value for controllable pitch propellers is 0.8. The warning is disregarded since its effect on the required comparison is small. It did reveal that the estimation of the diameter in the DDGx model gives bad prediction, and that a slightly bigger propeller diameter is needed: a bigger disk area decreases the loading and allows lower expanded area ratio. The second message was from the space module. As in the DDG51 match run, no space balance is reached. As shown in Table 2.2, for the DDG51, about 10000 ft^3 in additional volume are required for area balance. For the DDGx the required area is bigger than the available by approximately 14100 ft^3. The suspicious results of ASSET are thus ignored here as well.

6.2 Comparison between ASSET and DDGx Self-Balanced Model

The following sections compare the results of the Fortran model with those of ASSET, and analyze the differences between the weight-optimized ship and the DDG51. The comparison is presented in the order of the modules in ASSET.

6.2.1 Geometry

Table 6.2 shows the results for the geometrical information. In this case the differences between the model and ASSET are small, because the parameters are input to ASSET.

	DDG51	DDGx		
		ASSET	Model	Error [%]
L_{WL} [*ft*]	465.7	458.77	458.95	0.0
B [*ft*]	58.8	52.63	52.67	+0.1
T [*ft*]	20.1	16.43	16.83	+2.4
C_P	0.609	0.700	0.700	-
C_X	0.819	0.900	0.900	-
B/T	2.93	3.20	3.13	-2.2
L/B	7.92	8.72	8.71	+0.1
D_{10} [*ft*]	41.83	32.66	32.66	-
D_0 [*ft*]	52.06	45.45	45.45	-
D_{20} [*ft*]	36.29	34.19	34.19	-
C_W	0.787	0.835	0.863	+3.4
V_D [*ft*^{3}]	192009	239553	239553	-

Table 6.2. *Geometrical comparison.*

Given data in the ASSET model of the DDGx is underlined in the table. The small difference in the prediction of the draft for the DDGx is due to the different parent hull in ASSET. The difference in the water plane coefficient is understandable: while ASSET calculates this parameter based on the real offsets, the self-balanced model uses the empirical relation (2.39). As is expected, DDG51 is bigger and thus heavier than DDGx, although its hull is more slender. The higher C_P and C_X values make the DDGx fuller than the DDG51, but the DDG51 is fatter due to the lower L/B ratio. The price for the lower depth in the DDGx is the higher deckhouse volume. As shown in Table 6.6, the bigger superstructure decreases the stability indicator GM/B relative to the DDG51.

6.2.2 Resistance

Comparison of resistance module results is shown in Table 6.3. The wetted surface for all three cases includes the 1400 ft^2 wetted area of the sonar dome, which is considered as part of the hull in the resistance analysis.

	DDG51	DDGx		
		ASSET	Model	Error [%]
Endurance shaft horsepower [hp]	14731	17095.0	16618.3	-2.8
Wetted surface [ft^2]	30698	29443.0	29283.9	-0.5
Sustained speed [knt]	29.96	30.24	30.31	+0.2
Maximum speed [knt]	31.28	32.24	32.23	0.0

Table 6.3. *Resistance comparison.*

The correspondence between the results of ASSET and the model is excellent but not surprising, since they both employ identical algorithms. The small difference in the endurance shaft horsepower is due to the calibration of the DDGx model at sustained speed and not endurance speed. Figure 6.5 shows the behavior of frictional, residuary and total effective horsepower for the DDGx and the DDG51. Data is extracted from the resistance module.

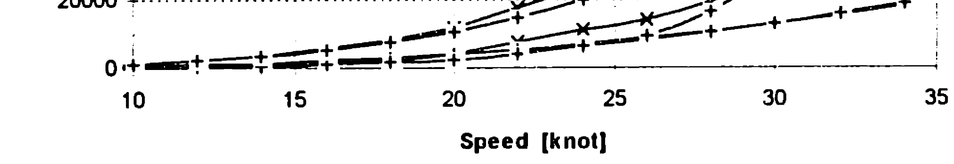


Figure 6.5. *Comparison of frictional, residuary and total effective horsepower.*

At endurance speed the DDG51 has less resistance, due to its lower C_P[2], but at high speeds, above approximately 28.5 knots, the trends change and the resistance is higher

[2] C_P has higher impact on residuary resistance at moderate speeds.

than for the DDGx. The change is caused due to an increase of the frictional and particularly residuary drag for the DDG51. The DDGx has thus higher sustained and maximum speeds.

The computation of the residuary drag involves the Worm curve factor. Motivated by the uncertainty of these values, additional residuary drag coefficient calculations were performed. The analysis included a different Worm curve, the 'DD calc' option in ASSET, recommended for SQS-53 sonar domes treated as part of the hull. It also examined calculations according to a regressive method performed at DTRC for destroyer-type hulls. The model was then synthesized, and the results are given in Table 6.4. For the regression methods one warning message was sent. The longitudinal center of buoyancy position *LCB* to L_{BL} ratio for this case is out of range.

Method	Worm=Original	Worm=DD calc	Regressive
Sustained speed [knt]	30.24	30.03	29.04
Maximum speed [knt]	32.24	31.81	30.08

Table 6.4. *Comparison of resistance results for different residuary drag predictive methods.*

The comparison reveals that the TSS resistance prediction may be optimistic. According to the regressive prediction method the optimal ship is not feasible due to sustained speed constraint violation. The results for the 'DD calc' Worm curve factors are close to that limit also. Figure 6.6 plots the residuary horsepower of the three methods for comparison.

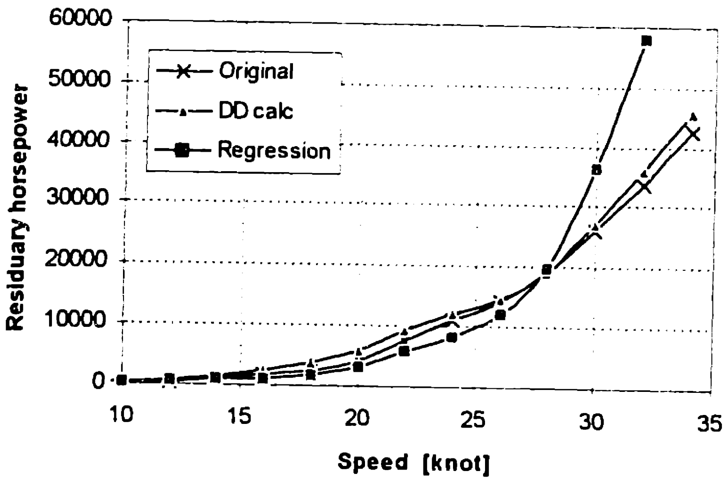


Figure 6.6. *DDGx residuary horsepower comparison between predictive methods.*

6.2.3 Electrical Load

The comparison for the electrical load analysis is provided in Table 6.5. As was already described in Section 2.7, due to historical reasons, it is assumed in the self-balanced model that the winter cruise condition and not the winter battle cruise condition requires the highest electrical load. In the DDG51, and also in this analysis, it was seen that this assumption is not true, but nevertheless close enough. Again there is good correlation for the DDGx. The higher electrical load associated with DDG51 is anticipated, due to her bigger volume: the empirical expressions used by the machinery module to compute electrical load depends mainly on internal volumes.

	DDG51	DDGx		
		ASSET	Model	Error [%]
Max marginal load [kW]	3439.0	3281.5	3341.5	+1.8
24 hours average load [kW]	2363.3	2145.4	2223.1	+3.6

Table 6.5. *Electrical load comparison.*

6.2.4 Weights and Stability

Table 6.6 shows comparative results for weight and stability parameters. The table combines stability and weight, because the former is driven strongly by the position of the center of gravity.

	DDG51	DDGx		
		ASSET	Model	Error [%]
Fuel weight [*lton*]	1186.7	1273.8	1218.9	-4.3
Light ship weight [*lton*]	6453.1	5686.6	5578.2	-1.9
Light ship VCG [*ft*]	25.0	20.93	21.46	+2.5
Full load weight [*lton*]	8126.9	7385.9	7288.8	-1.3
Full load VCG [*ft*]	22.69	19.33	19.97	+3.3
SWBS 100 weight [*lton*]	3115.5	2482.7	2346.1	-5.5
SWBS 200 weight [*lton*]	745.2	789.8	712.3	-9.8
SWBS 300 weight [*lton*]	315.1	306.1	314.0	+2.6
SWBS 400 weight [*lton*]	430.5	408.7	416.1	+1.8
SWBS 500 weight [*lton*]	824.2	734.3	778.2	+6.0
SWBS 600 weight [*lton*]	676.9	619.0	643.6	+4.0
Full load KB [*ft*]	11.94	9.18	9.37	+2.1
Full load BM [*ft*]	17.45	15.66	15.39	-1.7
Full load GM [*ft*]	5.98	4.79	4.79	0.0
GM/B	0.102	0.091	0.091	0.0

Table 6.6. *Weight and stability comparison.*

Excluding three categories, the agreement for the DDGx is within less then 5% error. The largest error is in SWBS 200 weight, nearly -10%. The discrepancies are not surprising, because ASSET performs detailed weight estimation of each component aboard the ship. The errors cancel each other, and the net error for the full load displacement is approximately 1%. The difference in the fuel weight is due to the difference in the endurance shaft horsepower discussed in Section 6.2.2. The DDG51 has higher weights in every aspect, related to the bigger hull, except for the fuel weight and SWBS 200. As discussed in Section 6.2.2, at cruise condition the DDG51 has lower resistance, and hence is required to carry almost 90 lton less fuel. The reduced SWBS 200 is not anticipated. An examination of the 3 digit SWBS 200 reports of DDG51 and DDGx shows that the main reason for the difference is the shafting[3], SWBS 240. The DDG51 also has better initial stability, not marginal as in the DDGx.

6.2.5 Space

The space calculations obtained from ASSET for both DDG51 and the current model give inconsistent results, and the match run for the DDG51 is not area-balanced. Therefore, the comparison in this category, given in Table 6.7, is made only for reference.

	DDG51	DDGx		
		ASSET	Model	Error [%]
Hull gross volume [ft^3]	800691.0	632417.0	671759.0	+6.2
Underwater volume [ft^3]	284441.5	258506.5	255108.0	-1.3
Above-water volume [ft^3]	516249.5	373910.5	416651.0	+11.4
Net required hull area [ft^2]	63573.0	55348.0	50352.2	-9.0
Net available hull area [ft^2]	48561.5	33328.6	37971.9	+13.9
Net required deckhouse area [ft^2]	14795.0	13301.0	10091.9	-24.1
Net available deckhouse area [ft^2]	20097.0	21213.0	22472.1	+5.9
Total net required area [ft^2]	78368.0	68649.0	60444.0	-12.0
Total net available area [ft^2]	68658.0	54542.0	60444.1	+10.8

Table 6.7. *Space Comparison.*

The key difference is in the two last rows, revealing the area balance. According to ASSET, the required area exceeds the available one by approximately 14100 ft^3. According to the self-balanced model, on the other hand, the space is exactly balanced. This is the result of the adjustment of the deckhouse volume made by the DDGx model. DDG51 values in Table 6.7 are higher than in the DDGx, due to the bigger hull. The difference in the above-water volume, the net available hull area and consequently the total net available area are due to the way the DDGx model computes the gross above-water

[3] Note that in Figure 6.3 one shaft is long, possibly too long. This fact is ignored, as the influence on the comparison is negligible.

hull volume, Equation (2.40). The value predicted by ASSET is calculated according to the actual body plan by the subdivision module.

6.2.6 Seakeeping

Lastly, seakeeping indices are compared. The self-balanced DDGx model does not evaluate seakeeping, and thus the results are only from the seakeeping module of ASSET. The results of this analysis are not considered in any way during the optimization process, and therefore are shown only for reference. The seakeeping module calculates two non-dimensional seakeeping indices, according to Bales and McCreight methods. Both are regressive predictions, evaluated in an effort to make a seakeeping performance assessment without the need to run a complicated computer code like the strip theory or panel methods. The results for the two ships are shown in Table 6.8.

	DDG51	DDGx
Bales	6.565	7.094
McCreight	14.220	12.928

Table 6.8. *Seakeeping comparison.*

Seakeeping behavior is considered better as the value of the indices increases. It can be seen that the Bales method predicts better performance for the DDGx, while McCreight gives the better score to DDG51. The numbers are close enough to conclude that with the certainty of this analysis, the two vessels have comparable seakeeping performance.

6.2.7 Comparative Naval Architecture

Comparative naval architecture allows a comparison to successful ships, and can reveal design lane deviations. A comparison of various naval architecture normalized parameters of the optimal DDGx and the existing DDG51 is shown in Table 6.9. The results for another existing ship, the FFG7, are also shown. This examination enables a comparison of specific features, which do not depend on the displacement or size of the vessel. The comparison analyzes weight, volume and power allocation and distribution. Data for the DDGx is taken from the ASSET model results. As can be seen, the normalized parameter of the DDGx are similar to those of the two in service ships.

In the weight allocation comparison, the machinery (W_2), electrical (W_3), control (W_4), armament (W_7) and ammunition (W_{F20}) system weight fractions of the DDGx are higher than for the other ships. The reason is the predefined HM&E and payload systems, identical to DDG51. In a real design the size and weight of SWBS200 and 300 would be lower, due to the smaller hull. The long shafts discussed in Section 6.2.4 also contribute to the ratio for the machinery weight. The low weight margin fraction relative to the FFG7 is due to the low user-defined weight margin factor, adopted for the DDGx from the DDG51 ASSET match run, as described in Section 2.9. The lighter hull of the DDGx relative to the DDG51 results in a lower lightweight to full load weight ratio, thus allowing allocation of more weight to loads. As a result, the fuel (W_{F41}), ammunition (W_{F20}) and overall load (W_{F00}) weight fractions are higher than in the DDG51.

	DDG51	DDGx	FFG7
Weight Allocation Fraction			
W_1/W_T	0.383	0.336	0.318
W_2/W_T	0.092	0.107	0.085
W_3/W_T	0.039	0.041	0.050
W_4/W_T	0.053	0.055	0.033
W_5/W_T	0.101	0.099	0.140
W_6/W_T	0.083	0.084	0.093
W_7/W_T	0.039	0.043	0.028
W_{M24}/W_T	0.004	0.004	0.019
W_{LS}/W_T	0.794	0.770	0.767
W_{F41}/W_T	0.146	0.172	0.145
W_{F20}/W_T	0.025	0.028	0.020
W_{F00}/W_T	0.206	0.230	0.233
Volume Allocation			
V_{HUW}/V_T	0.287	0.296	0.236
V_{H4W}/V_T	0.520	0.429	0.472
V_{HT}/V_T	0.807	0.725	0.708
V_D/V_T	0.193	0.275	0.291
V_{MB}/V_T	0.197	0.223	0.197
V_{F41}/V_T	0.051	0.062	0.064
Capacity Size Ratios			
P_I/W_T *[hp/lton]*	12.686	13.959	11.700
kW_I/W_T *[kW/lton]*	0.923	1.015	1.100
N_T/W_T *[men/lton]*	0.042	0.046	0.059
V_T/W_T *[ft3/lton]*	122.150	118.059	148.200
Specific Ratios			
W_1/V_T *[Lb/ft^3]*	7.030	6.378	4.810
W_2/P_I *[Lb/shp]*	16.191	17.160	16.180
V_{MB}/P_I *[ft^3/shp]*	1.896	1.885	2.500
W_3/kW_I *[Lb/kW]*	94.110	91.422	100.800
W_5/V_T *[Lb/ft^3]*	1.860	1.886	2.120
W_6/V_T *[Lb/ft^3]*	1.527	1.590	1.410
Densities			
W_T/V_T *[Lb/ft^3]*	18.338	18.974	15.110
W_{CS} *[lton]*	1012.700	991.200	352.200
W_{CS}/V_T *[Lb/ft^3]*	2.285	2.546	1.487
Overall			
W_{CS}/W_T	0.125	0.134	0.098
$W_{CS} \cdot V_S/W_T$	3.733	4.058	2.820

Table 6.9. *Comparative naval architecture ratios for the DDGx optimal ship.*

In the comparison of volume allocation, the differences are associated with the bigger - by 25% - superstructure of the DDGx. The higher machinery box volume fraction is a result of the approach used for the determination of the machinery spaces volume in Section 2.7. As recognized in Section 6.1, the volume of these may be higher than required.

The comparison of the capacity size ratios show that the "density" V_T/W_T of the DDGx is the lower, an expected result because weight optimization is conducted in this design. The ratio N_T/W_T is higher than in the DDG51 only due to the identical crew size. A real design would incorporate smaller crew.

The weight of the combat systems, W_{CS}, is calculated by

$$W_{CS} = W_4 + W_7 + W_{F21} + W_{F22} + W_{F23} + W_{F24} + W_{F42} \tag{6.2}$$

The small difference between the DDGx an the DDG51 is due to a small difference in W_4, the weight of control systems, apparently due to lower cabling weight. The reduced total volume of the DDGx gives her a higher combat system specific weight W_{CS}/V_T. Both ships have a significantly high ratio relative to the FFG7.

The last comparison shows similar trend for the weight portion of the combat systems, as expected. The last ratio, $W_{CS} \cdot V_S/W_T$, expresses the power of the war ship in an artificial way. The higher the combat system weight portion and sustained speed, the higher the power of the vessel. The DDGx offers the higher value.

6.3 Conclusions

The results obtained from ASSET for the optimal ship confirm that the analysis of the self-balanced model gives sufficient accuracy for a wide domain. The majority of results are very close to ASSET predictions, and, considering the simplicity and the time required to conduct the preliminary design with the self-balanced model the model is attractive in early design stages.

For this specific study, the comparison has verified the optimization results, for which it is possible to design a ship which satisfies the design requirements of the DDG51 class and is significantly lighter. The fuller hull shape of the DDGx allows the allocation of more area in the hull, while keeping the depth low in order to minimize the weight. The remaining required area is allocated in the deckhouse. The big deckhouse decreases the stability of the vessel, but the reduced *GM* to *B* lower limit, 0.09, enables the superstructure to be this big.

The reduced weight of the DDGx ship is not the only advantage relative to the DDG51. The cost which depends on the weight is lower also. The DDGx travels at sustained speed of 30.24 knots and maximum speed of 32.24 knots, faster than the DDG51. The DDGx has lower or at least equal signatures in all aspects: the visual and RCS signatures are lower due to the smaller dimension, the height in particular; the thermal, acoustic and electromagnetic signatures are at least as good as in the DDG51. The lower signatures impart the DDGx higher survivability and operational effectiveness. The latter expresses the chances of the vessel to complete its mission. The lower weight also allows operation in shallower water and narrower channels.

The evolutionary algorithm in this research, for the design and payload requirements, associates light ships with full-body hulls and high volume deckhouses. Typically, C_P and C_X values of Naval ships are lower than 0.64 and 0.85, respectively. The bounds are possibly due the feeling that fuller hulls have high resistance and consequently low sustained speed. In new designs the high values found here of 0.7 and 0.9, respectively, may not be considered. Possibly, as shown in Table 6.4 and Figure 6.6, the resistance prediction is not accurate due to incorrect Worm curve factors. For high speeds an extrapolation of Gertler's values is also performed. The extrapolation rule may also add some error to the calculation. When the other two predictive methods discussed in Section 6.2.2 are used, the optimal ship becomes non-feasible due to low sustained speed, which may verify the stigma of the designer.

Conclusions

The methodology adopted in this research is suggested for use. The optimization tool is constructed in the unconstrained solution space, in order to establish its reliability and to learn the trends. Once the robustness is proven, the algorithm does not care what is optimized, and the investigation in the constrained space is conducted. The collection of feasible solutions during the unconstrained runs gives an idea of the expected feasible values, which helps when deciding on the penalty intensity. The construction of the model does not involve evolutionary operators in the first phase. Different selection pressures and mutation rates are examined, and the best configuration is selected. The best configuration is the one that suffers from the least premature convergence. Then, enhanced evolutionary operators are incorporated, using the knowledge accumulated from previous runs. It is discovered that a non-uniform mutation operator is vital in any configuration, while other operators' contributions are not always better. The final configuration consists of non-uniform mutation, non-uniform selection pressure, and Baker's selection applied to the rank-based selection filter. It has an accuracy of 99.5%, and is suggested for use in other applications. The necessity of local tuning indicates that the initial required accuracy for all genes, but especially C_{ML}, C_{D10} and C_{BT}, is not sufficient, and contributes to the premature convergence. The conclusion is that the chromosomes best have a floating point number representation, which allows continuous exploration.

While the racist model does work, its tremendously long running time and other disadvantages make it non-comparable in all aspects to the method of penalty function, which is recommended. In case of highly constrained design problems, where penalties might be complicated to impose, the racist method may be useful.

The importance of a final local search, recommended by Holland [5], is also demonstrated. When a hill climbing technique is used, relatively low final results are improved significantly and premature convergence is almost completely eliminated. In this work separate programs are used for the evolutionary and hill climbing strategies, but it is recommended to integrate them together, as one optimization unit.

The remarkable reduction of more than 10% in weight from DDG51 design shows the effectiveness of a good optimization process. The research demonstrates the applicability of evolutionary optimization to ship design. The quality of the results predicted by the simplified model are confirmed by ASSET. The model, although simple, is better than anticipated. The study suggests the use of this basic tool for the "pre-concept" design stage. With the results obtained, ASSET is used to obtain a final concept design. The Mathcad model by itself is recommended for use in 13.412, for its reliable results, and the fact that it allows the student to concentrate on ship design rather than tediously struggle with the required data the current model requires.

The evolutionary algorithm has found that full vessels, with high prismatic and maximum section coefficients, and large deckhouses - within stability constraints - yield light ships. Usually such high values are not examined due to resistance considerations. However, additional resistance analysis must be performed. It is shown in Chapter 7 that a different resistance prediction method gives higher resistance, which makes the optimal ship non-feasible due to sustained speed constraint violation.

Future Study

As stated in Chapters 5 and 6, as well as in the conclusions, additional study is required in the resistance model. The convergence to high C_P and C_X values contradicts the shape of the majority of existing war ships. Improved accuracy can also achieved by estimation of propeller-hull interaction characteristics and modified propulsive coefficient prediction. There exists many regressive data in this area. The prediction of the propeller diameter requires a modification as well, as seen in Section 6.1.

Similarly, additional examination of the space module which determines the required area is needed. ASSET also seems to give poor prediction in this field.

The next step, but clearly the toughest one, is to incorporate the evolutionary optimization with a better design tool, ASSET for example.

The optimization conducted in this work is weight-based, but there are other applications for which the fitness function may differ. The tool will of course work for other functions, such as maximum speed. A comparative study of hulls of different optimal function may lead to enhanced hulls combining the best of those features together.

Better algorithm performance may be achieved by adjusting additional parameters such as the probability of crossover. I think the influence of this parameter has never been examined.

Additional consideration is needed for the limits of stability. In this work the lower limit is set to GM/B=0.09. The value is taken from the math model. ASSET, for example, allows the user to specify this limit. In the match run of DDG51 a value of 0.086 is used. As is discovered, light ships tend to have marginal stability. Thus, by allowing lower limits, lighter hulls can be obtained.

The model is limited in application because it allows only hull changes. Neither HM&E systems nor manning changes are included. An extended chromosome, including those parameters, would give wider exploration of possible solutions. This requires incorporation of additional modules for assessing the additional information. The more parameters the model has, the more accurate the results and more efficient the engineering judgment. Additional hull genes can be added as well, the flare angle for example. A more correct model should also account for different payload locations. The current model assumes that the payload components have a constant center of gravity position, relative to the baseline of the ship. For weight optimization the evaluation is conservative, but a more real prediction should relate the position with the location of the deck on which the components is installed. The best solution is to locate each payload item differently in the hull, as function of its dimensions, instead of using the total payload weight and center of gravity position.

Bibliography[1]

[1] Murphy, R. D., Sabat, D. J. and Taylor, R. J., "*Least Cost Ship Characteristics by Computer techniques*", Marine Technology, Vol. 2, 1965, pp. 174-202.

[2] Mandel, Philip and Leopold, Reuven, "*Optimization Methods Applied to Ship Design*", Transactions, The Society of Naval Architects and Marine Engineers, Vol. 74, 1966.

[3] Barrett, David S., "*Propulsive Efficiency of a Flexible Hull Underwater Vehicle*", Doctoral thesis, Massachusetts Institute of Technology, Department of Ocean Engineering, May 1996.

[4] Bolc, Leonard and Cytowski, Jerzy, "*Search methods for artificial intelligence*", Academic Press, 1992.

[5] Holland, John H., "*Adaptation in Nature and Artificial Systems*", The MIT press, 1992.

[6] Michalewicz, Zbigniew, "*Genetic Algorithms + Data structures = Evolution Programs*", Springer-Verlag, 1992.

[7] Goldberg, David E., "*Genetic Algorithms in Search, Optimization, and Machine Learning*", Addison-Wesley Publishing Company, 1989.

[8] Svirezhev, Y. M. and Passekov, V. P., "*Fundamentals of Mathematical Evolutionary Genetics*", Kluwer Academic Publishers, Mathematics and its Applications, Volume 22, London, 1989.

[9] Davis L. (editor), "*Genetic Algorithms and Simulated Annealing*", Morgan Kaufmann Publishers inc., 1987.

[1] References are listed according to their appearance in the text.

[10]Laarhoven, P. J. M. van and Aarts, E. H. L., "*Simulated Annealing: Theory and Applications*", D. Reidel, Dordrecht, Holland, 1987.

[11]Stender, J., Hillebrand, E. and Kingdon, J. (editors), "*Genetic Algorithms in Optimization, Simulation and Modeling*", IOS Press, 1994.

[12]Holland, John and Goldberg, David, "*Genetic Algorithms and Machine Learning*", Machine Learning, Volume 3, 1988.

[13]DeJond, K. A., "*Genetic Algorithm Based Learning*", Machine Learning: an Artificial Intelligence Approach, Morgan Kaufmann Publishers, Volume 3, 1990.

[14]Reed, Michael R., "*Ship Synthesis Model for Naval Surface Ships*", Master thesis, Massachusetts Institute of Technology, Department of Ocean Engineering, May 1976.

[15]Gillmer, Thomas C. and Johnson, Bruce, "*Introduction to Naval Architecture*", Naval Institute Press, 3rd printing, 1987.

[16]Lewis, Edward V. (editor), "*Principles of Naval Architecture Second Revision*", Volume I, The Society of Naval Architects and Marine Engineers, 1988.

[17]ASSET/MONOSC, "*User Manual for the Hull generation Module*", Version 3.0, David Taylor Model Basin, Carderock Division, Naval surface Weapons Center, May 1990

[18]DDS 051-1, "*Prediction of Smooth Water Powering Performance for Surface Displacement Ships*", Naval Sea System Command, 15 May 1984.

[19]Taylor, D. W., "*The Speed and Power of Ships*", Third edition, U.S. Government Printing Office, 1943.

[20]Gertler, Morton, "*A Reanalysis of the Original Test Data for the Taylor Series*", Report 806, Navy Department, David W. Taylor Model Basin, March 1954.

[21]ASSET/MONOSC, "*User Manual for the Resistance Module*", Version 3.3, David Taylor Model Basin, Carderock Division, Naval surface Weapons Center, Aug 1992.

[22]"*Residuary Resistance Data for U.S. Navy Destroyer Type Hull Forms*", NAVSEA report c-3213-80-27, December 1980.

[23]ASSET/MONOSC, "*User Manual for the Initialization Module*", Version 3.0, David Taylor Model Basin, Carderock Division, Naval surface Weapons Center, May 1990.

[24]Woodward, John B., "*Propulsion by the Internal Combustion Engine*", University of Michigan, Department of Naval Architecture and Marine Engineering, Report #122, October 1971.

[25]ASSET/MONOSC, "*User Manual for the Machinery Module*", Version 3.1, David Taylor Model Basin, Carderock Division, Naval surface Weapons Center, September 1990.

[26]Brent, R. P. (1971), "*An Algorithm with Guaranteed Convergence for Finding a Zero of a Function*", The Computer Journal, 14, 422-425.

[27]Gerald, C. F. and Wheatley, P. O., "*Applied Numerical Analysis*", 4th edition, Addison Wesley, 1989.

[28]DDS 079-2, "Minimum Required Freeboard for U. S. Naval Surface Ships", NAVSEA, 1982.

[29]DDS 200-1, "*Calculation of Surface Ship Endurance Fuel Requirements*", Naval Sea System Command, 1 March 1982.

[30] ASSET/MONOSC, "*User Manual for the Space Module*", Version 3.0, David Taylor Model Basin, Carderock Division, Naval surface Weapons Center, May 1990.

[31]ASSET/MONOSC, "*User Manual for the Deckhouse Module*", Version 3.0, David Taylor Model Basin, Carderock Division, Naval surface Weapons Center, May 1990.

[32]ASSET/MONOSC, "*User Manual for the Weight Module*", Version 3.0, David Taylor Model Basin, Carderock Division, Naval surface Weapons Center, May 1990.

[33]DDS 079-1, "Stability and Buoyancy of U.S. Naval Surface Ships", Department of the Navy, Naval Ship Engineering Center, 1 August 1975.

[34]Muller, D. E., "*A method for solving algebraic equations using an automatic computer*", Mathematical Tables and Aids to Computation, 10, 208-215, 1956.

[35]Michalewicz, Z. and Janikow, C., "*Handling Constraints in Genetic Algorithms*", in "*Proceedings of the Fourth International Conference on Genetic Algorithms*", Morgan Kaufmann Publishers, Los Altos, CA, 1991, pp. 151-157.

[36]DeJong, K. A., "*An Analysis of the Behavior of a Class of Genetic Adaptive Systems*", Doctoral Dissertation, University of Michigan, 1975.

[37]Baker, J. E., "*Reducing Bias and Inefficiency in the Selection Algorithm*", pp. 14-21 in "*Proceedings of the Second International Conference on Genetic Algorithms*", Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[38]Eshelman, L. J., Caruana, R. A., and Schaffer, J. D., "*Biases in the Crossover Landscape*", pp. 10-19 in "*Proceedings of the Third International Conference on Genetic Algorithms*", Morgan Kaufmann Publishers, Los Altos, CA, 1989.

[39]"*Mathcad User's guide*", Mathsoft inc., Cambridge, 1996.

[40]Hahn, Brian, "*Fortran 90 for Scientists and Engineers*", Chapman & Hall, March 1994.

Appendix A

Ship Design Nomenclature

γ_F	Fuel specific volume.
γ_F	Potable water specific volume.
γ_{HF}	Helicopter fuel specific volume.
γ_{LO}	Lubrication oil specific volume.
η	Transmission efficiency.
ν_{SW}	Salt water dynamic viscosity.
ρ_A	Air specific weight.
ρ_{DH}	Deckhouse specific weight.
ρ_{SW}	Salt water specific weight.
A_0	Empirical coefficient for the calculation of C_{STSS}.
A_1	Empirical coefficient for the calculation of C_{STSS}.
A_2	Empirical coefficient for the calculation of C_{STSS}.
A_{COXO}	COXO living deck area at deckhouse.
A_{DA}	Available deckhouse deck area.
A_{DB}	Bridge and chart room deck area.
A_{DIE}	Engine inlet/exhaust deck area at deckhouse.
A_{DL}	Total living deck area at deckhouse.
A_{DM}	Maintenance deckhouse deck area.
A_{DO}	Officers living space deck area at deckhouse.
A_{DPA}	Required armament payload deckhouse deck area.
A_{DPC}	Required C&D payload deckhouse deck area.
A_{DPR}	Deckhouse payload area.
A_{DR}	Total deckhouse required area.
A_{eIE}	Required inlet/exhaust cross section area per generator engine.
A_{GIE}	Total required inlet/exhaust cross section area for generator engines.
A_{HA}	Available hull deck area.
A_{HAB}	Hull or deckhouse habitability deck area per crew member.
A_{HIE}	Engine inlet/exhaust deck area at hull.
A_{HL}	Hull or deckhouse living deck required area.

A_{HPA}	Required armament payload hull or deckhouse deck area.
A_{HPC}	Required C&D payload hull or deckhouse deck area.
A_{HPR}	Hull or deckhouse payload area.
A_{HR}	Total hull or deckhouse required area.
A_{HS}	Hull or deckhouse stores deck area.
A_{HSF}	Ship functions deck area at hull or deckhouse.
A_{IE}	Required inlet/exhaust cross section area per propulsion engine.
A_{PIE}	Total required inlet/exhaust cross section area for propulsion engines.
A_{PRO}	Projected area of above water hull.
A_{SD}	Required sonar dome/appendages area.
A_{TA}	Available ship deck area.
A_{TR}	Total ship required area.
A_{W}	Frontal area of ship (above water).
B	Beam on waterline.
BAL_{TYP}	Type of ballast system (compensated/uncompensated).
BM	Distance between center of buoyancy and metacenter.
$C_{\Delta L}$	Displacement to length ratio.
C_{A}	Roughness frictional drag correction coefficient (correlation allowance).
C_{AA}	Air drag coefficient.
C_{BT}	Beam to draft ratio.
C_{D10}	Length to depth ratio.
C_{DAPP}	Appendage drag coefficient.
C_{DHMAT}	Deckhouse material coefficient.
C_{F}	Frictional drag coefficient.
C_{GMB}	GM to Beam ratio.
C_{HMAT}	Hull material coefficient.
C_{IT}	Transverse water plane inertia coefficient.
C_{LB}	Length to beam ratio.
CN	Cubic number.
C_{P}	Prismatic coefficient.
C_{PROPD}	Propeller area coefficient.
$C_{R2.25}$	Gertler residuary drag coefficient for 2.25 beam to draft ratio.
$C_{R3.00}$	Gertler residuary drag coefficient for 3.00 beam to draft ratio.
$C_{R3.75}$	Gertler residuary drag coefficient for 3.75 beam to draft ratio.
C_{RD}	Raised deck coefficient.
C_{RTSS}	TSS residuary drag coefficient.
C_{STSS}	TSS wetted surface coefficient.
C_{V}	Volumetric coefficient.
C_{W}	Waterplane coefficient.
C_{X}	Maximum cross section coefficient.
D_{0}	Depth at station 0.
D_{0MIN}	Minimum depth at station 0.
D_{10}	Hull depth at station 10.
D_{10MIN}	Minimum depth at station 10.
D_{20}	Depth at station 20.

D_{20MIN}	Minimum depth at station 20.
D_{AV}	Average depth.
D_P	Propeller diameter.
E	Endurance range.
$E24MF$	24 hour electrical load margin factor.
$EDMF$	Electrical load design margin factor.
$EFMF$	Electrical load future growth margin factor.
EHP	Design ship effective horsepower required.
ERR	Weight balance error.
ERR_A	Area balance error.
ERR_V	Sustained speed balance error.
F_0	Freeboard at station 0.
f_l	Fuel rate margin for propulsion engines.
F_{10}	Freeboard at station 10.
f_{le}	Fuel rate margin for generator engines.
F_{20}	Freeboard at station 20.
F_{AV}	Average freeboard.
FF	Form factor.
F_P	Payload fraction.
FR_{AVG}	Average propulsion engines fuel consumption at endurance speed.
FR_{GAVG}	24 hour average generator engines fuel consumption.
FR_{GSP}	24 hour average specified generator fuel rate.
FR_{SP}	Specified propulsion engines fuel rate at endurance speed.
f_S	Shaft length coefficient.
GM	Distance between center of gravity and metacenter.
H_{DK}	Average deck height.
H_{MB}	Machinery box height.
H_{MBMIN}	Minimal machinery box height.
H_R	Raised deck height.
KB	Vertical center of buoyancy position.
KG	Vertical center of gravity position.
KG_{LS}	Lightship KG.
KG_{MARG}	KG margin.
KW_{24}	24 hour average electrical load.
KW_{24AVG}	24 hour average electrical load with design margin.
KW_A	Miscellaneous auxiliary machinery electrical load.
KW_{AC}	Air conditioning electrical load.
KW_B	Auxiliary boiler electrical load.
KW_{CPS}	CPS electrical load.
KW_E	Electrical plant electrical load.
KW_F	Firemain electrical load.
KW_{fins}	Electric power requirement for fin stabilizers.
KW_G	Nominal service generator electric power.
KW_{GREQ}	Electrical power required per generator.
KW_H	Heating electrical load.

KW_{HN}	Fuel handling electrical load.
KW_{M}	Miscellaneous electrical load.
KW_{MFL}	Maximum functional electrical load.
KW_{MFLM}	Maximum functional electrical load with design and growth margins.
KW_{NP}	Non-payload functional electrical load.
KW_{P}	Propulsion electrical load.
KW_{PAY}	Payload electric power required.
KW_{S}	Steering electrical load.
KW_{SERV}	Services and work spaces electrical load.
KW_{V}	Ventilation electrical load.
L_{WL}	Length on waterline.
N_{A}	Number of additional accommodations.
N_{DIE}	Number of deckhouse decks impacted by propulsion/generator inlet/exhaust.
N_{E}	Number of enlisted.
N_{fins}	Number of roll fins pairs.
N_{G}	Number of ship service generators.
N_{HelE}	Number of hull decks impacted by generator engine inlet/exhaust.
N_{HPIE}	Number of hull decks impacted by propulsion engine inlet/exhaust.
N_{O}	Number of officers.
N_{P}	Number of propellers.
N_{PENG}	Number of propulsion engines.
N_{T}	Crew size.
P_{BMAX}	Maximum continuos brake horsepower (all propulsion engines together).
P_{BPENG}	Nominal propulsion engine brake horsepower.
PC	Propulsive coefficient.
P_{EAA}	Air effective horsepower.
P_{EAPP}	Appendage effective horsepower.
P_{eBAVG}	Average endurance brake horsepower required.
P_{EBH}	Bare hull effective horsepower.
P_{Efins}	Effective horsepower of roll fins.
P_{ET}	Total effective horsepower.
P_{G24}	24 hour average power of ship service engines.
P_{I}	Actual installed horsepower.
P_{IREQ}	Installed horsepower required.
PMF	Power margin factor.
$PROP_{TYP}$	Type of propeller (FP/CPP).
P_{WG}	Lightship vertical moment.
P_{WGL}	Variable load vertical moment.
P_{X}	Vertical moment dummy parameter.
R	Speed to length ratio.
R_{F}	Frictional resistance.
r_{nPE}	Endurance rpm to maximum rpm ratio for propulsion engines.
r_{nG}	24 hour average rpm to maximum rpm ratio for ship service engines.
R_{N}	Reynolds number.

r_{pPE}	Endurance bhp to maximum bhp ratio for propulsion engines.
r_{pG}	24 hour average bhp to maximum bhp ratio for ship service engines.
r_{QPE}	Endurance torque to maximum torque ratio for propulsion engines.
r_{QG}	24 hour average torque to maximum torque ratio for ship service engines.
R_R	Residuary resistance.
R_{RTSS}	TSS residuary resistance.
R_T	Total bare hull resistance.
SFC_{ePE}	Specific fuel rate for propulsion engines at endurance speed.
SFC_G	Specific fuel rate for ship service engines at maximum continuous rate.
SFC_{G24}	24 hour average fuel rate for ship service engines (per hp).
SFC_{GE24}	24 hour average fuel rate for ship service engines (per kW).
SFC_{PE}	Specific fuel rate for propulsion engines at maximum continuous rate.
SHP	Shaft horsepower.
SHP_e	Endurance shaft horsepower.
SHP_S	Sustained speed shaft horsepower.
SON_{TYP}	Type of installed sonar (SQS-53/SQS-56/none)
S_S	Wetted surface area.
S_{SD}	Sonar dome wetted surface area.
S_{TSS}	TSS wetted surface area.
T	Draft.
TPA	Tailpipe allowance for tanks.
T_S	Stores period.
V_{AUX}	Auxiliary machinery rooms volume.
V_{BAL}	Clean ballast tank volume (for trim/heel adjustments).
VCG_{100}	Vertical center of gravity of ship's structure.
VCG_{171}	Vertical center of gravity of masts.
VCG_{180}	Vertical center of gravity of mechanical foundations.
VCG_{200}	Vertical center of gravity of ship's propulsion plant.
VCG_{237}	Vertical center of gravity of APU.
VCG_{300}	Vertical center of gravity of ship's electrical plant.
VCG_{400}	Vertical center of gravity of command and control systems.
VCG_{498}	Vertical center of gravity of sonar dome's water.
VCG_{500}	Vertical center of gravity of auxiliary systems.
VCG_{517}	Vertical center of gravity of auxiliary steam system.
VCG_{600}	Vertical center of gravity of outfit and furnishing.
VCG_{700}	Vertical center of gravity of armament.
VCG_{AUX}	Vertical center of gravity of auxiliary machinery.
VCG_{BH}	Vertical center of gravity of bare hull.
VCG_{BM}	Vertical center of gravity of basic machinery.
VCG_{CC}	Vertical center of gravity of cabling.
VCG_{CO}	Vertical center of gravity of miscellaneous command and control systems.
VCG_{DH}	Vertical center of gravity of deckhouse.
VCG_{F10}	Vertical center of gravity of crew.
VCG_{F31}	Vertical center of gravity of provisions.
VCG_{F32}	Vertical center of gravity of stores.

VCG_{F41}	Vertical center of gravity of fuel.
VCG_{F46}	Vertical center of gravity of lubrication oil.
VCG_{F52}	Vertical center of gravity of potable water.
VCG_{IC}	Vertical center of gravity of communication/gyro/navigation systems.
VCG_{L}	Variable load vertical center of gravity.
VCG_{LS}	Lightship vertical center of gravity.
VCG_{OFH}	Vertical center of gravity of hull fittings.
VCG_{OFP}	Vertical center of gravity of personnel related outfit.
VCG_{P}	Payload vertical center of gravity.
VCG_{P100}	Structural payload vertical center of gravity.
VCG_{P400}	Command and control payload vertical center of gravity.
VCG_{P500}	Auxiliary systems payload vertical center of gravity.
VCG_{P600}	Outfit and furnishing payload vertical center of gravity.
VCG_{ST}	Vertical center of gravity of shafting.
VCG_{VP}	Variable payload vertical center of gravity.
V_{D}	Deckhouse volume.
V_{DMAX}	Deckhouse volume upper limit.
V_{e}	Endurance speed.
V_{F}	Total fuel tanks volume.
V_{fe}	Volume of required fuel for generators.
V_{FL}	Underwater hull volume.
V_{FP}	Volume of required fuel for propulsion engines.
V_{HA}	Available hull volume.
V_{HAW}	Above water hull volume.
V_{HF}	Helicopter fuel tanks volume.
V_{HL}	Lost hull volume due to raised deck.
V_{HR}	Total hull or deckhouse required volume.
V_{HT}	Total hull volume.
V_{HUW}	Under water hull volume.
V_{LO}	Lubrication oil tanks volume.
V_{MAX}	Maximum speed.
V_{MB}	Machinery box volume.
V_{S}	Sustained speed.
V_{SEW}	Sewage tanks volume.
V_{T}	Total ship volume.
V_{TA}	Available ship volume.
V_{TK}	Total ship tanks volume.
V_{TR}	Total ship required volume.
V_{W}	Potable water tanks volume.
V_{WASTE}	Waste oil tanks volume.
W_{1}	Structure weight (SWBS 100).
W_{171}	Masts weight.
W_{180}	Foundations weight.
W_{2}	Propulsion plant weight (SWBS 200).
W_{237}	APU weight.

W_3	Electrical plant weight (SWBS 300).
W_4	Command/control/surveillance weight (SWBS 400).
W_{498}	Sonar dome water weight.
W_5	Auxiliary systems weight (SWBS 500).
W_{593}	Environmental support weight.
W_{598}	Auxiliary systems operating fluids weight.
W_6	Outfit and furnishing weight (SWBS 600).
W_7	Armament weight (SWBS 700).
W_{AUX}	Auxiliary machinery weight.
W_B	Bearings weight.
W_{Be}	Burnable electrical endurance fuel weight.
W_{BH}	Bare hull weight.
W_{BM}	Basic machinery weight.
W_{BP}	Burnable propulsion endurance fuel weight.
W_{CC}	Cabling weight.
WCF	Worm curve interpolated correction coefficient.
W_{CO}	Miscellaneous SWBS 400 weight.
W_{CPS}	CPS weight.
W_{CS}	Combat system weight.
W_{DH}	Deckhouse weight.
W_{F10}	Crew weight.
W_{F31}	Provisions weight.
W_{F32}	General stores weight.
W_{F41}	Total fuel weight.
W_{F42}	Helicopter's fuel weight.
W_{F46}	Lubrication oil weight.
W_{F52}	Potable water weight.
W_{Fe}	Weight of required generator fuel.
W_{FL}	Full load displacement.
W_{FL1}	Full load displacement (*guessed value, for first iteration only*).
W_{FP}	Weight of required propulsion fuel.
W_{IC}	Gyro/internal communication/navigation weight.
W_L	Variable load weight.
W_{LS}	Lightship weight.
W_{M24}	Weight margin (for future growth).
WMF	Lightweight margin factor.
W_{OFH}	Hull fittings weight.
W_{OFP}	Personnel related outfit and furnishing weight.
W_P	Total military payload weight.
W_{P100}	Structure payload weight.
W_{P400}	Command and surveillance payload weight.
W_{P500}	Mission handling/support payload weight.
W_{P600}	Mission outfit payload weight.
W_{PR}	Propellers weight.
W_S	Shafts weight.

W_{ST}	Total shafting weight.
W_{T}	Total ship weight.
W_{VP}	Variable military payload weight.

Appendix B

Evolutionary Methods Nomenclature

$\delta(S)$	Defining length of schema S.
$o(S)$	Order of schema S.
$\xi(t,S)$	Number of chromosomes at generation t matched by the schema S.
a	Arithmetical crossover parameter.
a_i	Lower boundary of ith gene.
b_i	Upper boundary of ith gene.
b	Mutation non-uniformity degree.
CS	Center of selection.
ERR_x	Amount of xth constraint violation in percent.
$\bar{F}(t)$	Average fitness of entire population at generation t.
$F(t,S)$	Average fitness of all ξ chromosomes.
$fitness(x_i,t)$	Fitness function of the ith chromosome at generation t.
gen	Generation number.
gen_max	Maximum allowable number of generations.
k_i	Required accuracy (in decimal points) of ith gene.
m	Number of genes or bits in a chromosome.
$n\_converge$	Number of successive identical results required to declare convergence.
N_i	Number of divisions of the domain $[a_i, b_i]$.
q_i	Cumulative probability of survival of ith chromosome.
P_i	Probability of survival of ith chromosome.
P_C	Crossover probability.
P_M	Mutation probability.
$penalty$	Penalty value for non-feasible solution.
pop_size	Population size.
P_S	Selection pressure.
P_{UP}	Mutation update probability.
$span(t)$	Standard deviation of entire population at generation t.

Appendix C

DDGx payload and adjustments table

DDGx payload and adjustments table is taken entirely from the ASSET match run of DDG51. All data is obtained from the weight and summary modules. The items are classified by the SWBS system. Since ASSET algorithms are empirical, they are not guaranteed to give the actual results of the DDG51 in all cases, and thus inaccurate results are corrected to reflect the actual results. These corrections are called adjustments: there is no linkage between them and the military payload, although both are entered via the same table in ASSET. Adjustments in this appendix are shadowed. Since the DDGx model is calibrated to provide DDG51 actual results when her hull parameters are employed, the adjustments are not needed. The calibration includes the influence of the adjustments. Therefore, in this appendix, two weight summations are introduced: payload and adjustments sum, and payload sum. The first, denoted by '*PA*', includes adjustments data, while the latter, marked '*P*', does not. Only payload data is used for the design requirements in Section 2.1. Area requirements are listed with their SSCS[1] code for reference only. The codes are not used by the model. The requirements are divided into deckhouse and hull. Deckhouse area is an area which must be allocated in the superstructure; Hull area can be allocated in the hull or the deckhouse. Electrical loads are given for winter cruise and winter battle cruise conditions. However, only the first are used in the model, see Section 2.7. The variable payload, at the end of the table, is identical to group F00, which includes ammunition and helicopter fuel, equipment which is not fixed in the ship and requires replenishment.

All calculations are performed in an Excel worksheet. Not that they are complicated, but it enables quick changes, and creates a template, for future use for other vessels.

[1] Shipboard Space Classification System

Name	SWBS	Weight [lton]	VCG [ft]	Moment [lton*ft]	Area key	Hull area [ft^2]	Dkhs area [ft^2]	Cruise [kW]	Battle [kW]
Shell plating	111	69.9	-15.9	-1111.4	None	0.0	0.0	0.0	0.0
Inner bottom	113	47.3	6.2	293.3	None	0.0	0.0	0.0	0.0
Stanchions	115	9.9	29.5	292.1	None	0.0	0.0	0.0	0.0
Longitudinal framing	116	11.2	145.6	1630.7	None	0.0	0.0	0.0	0.0
Transverse framing	117	23.2	-14.7	-341.0	None	0.0	0.0	0.0	0.0
Longitudinal bulkheads	121	49.9	28.5	1422.2	None	0.0	0.0	0.0	0.0
Transverse bulkheads	122	45.2	26.2	1184.2	None	0.0	0.0	0.0	0.0
Trunks + enclosures	123	28.1	13.3	373.7	None	0.0	0.0	0.0	0.0
Main deck	131	20.7	32.6	674.8	None	0.0	0.0	0.0	0.0
01 level deck	136	53.3	41.1	2190.6	None	0.0	0.0	0.0	0.0
First platform	141	26.8	24.2	648.6	None	0.0	0.0	0.0	0.0
Second platform	142	33.8	14.4	486.7	None	0.0	0.0	0.0	0.0
Inner bottom portion modeled as platform	143	-3.2	7.1	-22.7	None	0.0	0.0	0.0	0.0
MK45 5"/54 gun HY-80 armor (level II)	164	20.2	38.3	773.7	None	0.0	0.0	0.0	0.0
29 cell VLS armor - level III HY-80	164	14.0	33.3	466.2	None	0.0	0.0	0.0	0.0
61 cell VLS armor - level III HY-80	164	21.1	33.3	702.6	None	0.0	0.0	0.0	0.0
Unidentified ballistic plating	164	25.9	50.6	1310.5	None	0.0	0.0	0.0	0.0
Group 100	*WPA100*	497.3	22.1	10974.7		0.0	0.0	0.0	0.0
	WP100	81.2	40.1	3253.0		0.0	0.0	0.0	0.0
CIC W/12x UYQ-44 & 2x LSD	411	17.3	39.7	686.8	A1131	1989.0	0.0	74.5	75.5
Navigation system	420	0.0	57.9	0.0	None	0.0	0.0	16.4	20.5
External communication	440	32.4	33.9	1098.4	A1111	1270.0	95.0	93.3	96.4
SPS-67 surface search radar	451	1.7	71.5	121.6	A1121	0.0	70.0	8.0	0.0
MK XII AIMS IFF	455	2.3	71.1	163.5	None	0.0	0.0	3.2	4.0
SPY-1D MFAR - single transmitter	456	54.3	56.3	3057.1	A1121	0.0	1594.0	269.0	474.3
SQQ-28 LAMPS MKIII electronics	460	3.5	44.8	156.8	None	0.0	0.0	5.3	5.5
SQR-19 TACTAS (elex in hull sonar)	462	23.3	34.3	799.2	A1122	473.0	0.0	26.6	26.6
SLQ-32 (V2) passive ECM	472	3.0	63.3	189.9	A1141	40.0	132.0	6.4	6.4
AN/SLQ-25 NIXIE	473	3.6	34.3	123.5	A1142	172.0	0.0	3.0	4.2
SLQ-32 (V2) MK36 DLS w/4 launchers	474	1.1	55.4	60.9	None	0.0	0.0	2.4	2.4
AEGIS based 5" GFCS (UYQ-21 UYQ-44)	481	0.7	35.8	25.1	A1210	0.0	150.0	12.3	42.7
MK16 CIWS weapon control system	481	1.0	56.3	56.3	A1210	0.0	464.0	3.2	10.4
Tomahawk weapon control system (in CSER)	482	5.6	34.0	190.4	None	0.0	0.0	11.5	11.5
MK99 GMFCS w/3 SPG-62 illuminator	482	14.3	62.7	896.6	A1220	0.0	959.0	34.7	65.2
AN/SWG-1 Harpoon luanch control system in CIC	482	1.1	36.0	39.6	None	0.0	0.0	12.1	12.1
VLS weapon control system	482	0.7	34.5	24.2	A1220	56.0	0.0	15.0	18.0
VLS weapon control system	482	0.7	34.5	24.2	A1220	56.0	0.0	15.0	18.0
ASW control system (ASWCS)	483	4.8	30.8	147.8	None	0.0	0.0	19.5	19.5
MK3 integrated FCS	484	5.1	32.8	167.3	None	0.0	0.0	3.0	3.0
Operational readiness test system	491	4.6	44.3	203.8	None	0.0	0.0	12.0	12.0
Shipboard NON-TACT auto process II	493	4.0	30.7	122.8	A1131	112.0	0.0	0.7	0.7
CIC - CSER area	None	0.0	0.0	0.0	A1131	2236.0	0.0	0.0	0.0
Group 400	*WPA400*	185.1	45.1	8355.6		6404.0	3464.0	647.1	928.9
	WP400	176.5	45.5	8029.0		6292.0	3464.0	618.0	895.7
29 cell magazine dewatering system	529	1.5	32.5	48.8	None	0.0	0.0	0.0	0.0
61 cell magazine dewatering system	529	3.0	32.5	97.5	None	0.0	0.0	0.0	0.0
Cooling system delta	532	4.0	10.8	43.2	None	0.0	0.0	8.0	8.0
SPY-1D MFAR - equipment cooling delta	532	9.0	10.8	97.2	None	0.0	0.0	0.0	0.0
LAMPS MKIII: helo in flight refuel system	542	7.6	27.3	207.5	A1380	44.0	0.0	1.3	1.3
LAMPS MKIII: helo securing system	588	3.6	42.1	151.6	None	0.0	0.0	0.0	0.0
Group 500	*WPA500*	28.7	22.5	645.7		44.0	0.0	9.3	9.3
	WP500	15.7	32.2	505.3		44.0	0.0	1.3	1.3
MK45 5"/54 gun (hand SD)	710	50.0	46.1	2305.0	A1210	285.0	0.0	36.6	50.2
2x MK16 20mm CIWS & workshops	711	13.2	62.8	829.0	A1210	0.0	321.0	14.0	42.0
2x Harpoon SSM QUAD HCLS (LVL II hardened)	720	5.4	44.5	240.3	None	0.0	0.0	0.0	0.0
4x MK41 VLS 29 cell w/3 Tomahawk + 23 SM-2 + 3 VLASROC	721	82.8	31.5	2608.2	A1220	17.0	0.0	31.1	31.1
8x MK41 VLS 61 cell w/7 Tomahawk + 49 SM-2 + 5 VLASROC	721	147.8	30.5	4507.9	A1220	128.0	0.0	63.4	63.4
VLS weapons handling	722	1.0	38.3	38.3	A1220	75.0	0.0	0.0	0.0
VLS weapons handling	722	1.0	38.3	38.3	A1220	75.0	0.0	0.0	0.0
2x MK32 SVTT on deck	750	2.7	45.3	122.3	None	0.0	0.0	0.6	1.1
Small arms and pyro stowage	760	8.0	36.2	289.6	A1900	203.0	0.0	0.0	0.0
LAMPS MKIII: helicopter REARM + magazine	780	2.7	48.4	130.7	A1374	212.0	0.0	0.0	4.4
Group 700	*WPA700*	314.6	35.3	11109.6		995.0	321.0	145.7	192.2
	WP700	314.6	35.3	11109.6		995.0	321.0	145.7	192.2

Table C.1. *Payload and adjustments table.*

Name	SWBS	Weight [lton]	VCG [ft]	Moment [lton*ft]	Area key	Hull area [ft^2]	Dkhs area [ft^2]	Cruise [kW]	Battle [kW]
MK36 DLS SRBOC cannisters - 100 rds	F21	2.2	55.4	121.9	None	0.0	0.0	0.0	0.0
MK16 20mm CIWS Ammunition - 16000 rds	F21	8.4	61.8	519.1	A1210	0.0	257.0	0.0	0.0
MK45 5"/54 ammunition - 600 rds	F21	33.1	17.9	592.5	A1210	705.0	0.0	0.0	0.0
Harpoon missiles - 8 rds	F21	15.8	48.0	758.4	None	0.0	0.0	0.0	0.0
Missiles - 3 Tomahawk + 23 SM-2 + 3 VLASROC	F21	44.2	34.1	1507.2	A1220	1289.0	0.0	0.0	0.0
Missiles - 7 Tomahawk + 49 SM-2 + 5 VLASROC	F21	93.8	33.1	3104.8	A1220	2140.0	0.0	0.0	0.0
MK46 lightweight ASW torpedoes - 6 rds	F21	1.4	45.3	63.4	None	0.0	0.0	0.0	0.0
Small arms ammunition - 7.62 mm + 50 cal + pyro	F21	4.1	35.8	146.8	None	0.0	0.0	0.0	0.0
Bathythermograph probes	F29	0.2	25.7	5.1	None	0.0	0.0	0.0	0.0
LAMPS MKIII: aviation fuel (JP-5)	F42	64.4	13.0	837.2	None	0.0	0.0	0.0	0.0
Sea water in compensating tanks	F51	67.2	3.9	262.1	None	0.0	0.0	0.0	0.0
Correction to fresh water GRS	F52	-10.1	5.8	-58.6	None	0.0	0.0	0.0	0.0
Group F00	*WFPA*	324.7	24.2	7859.9		4134.0	257.0	0.0	0.0
	WFP	334.8	23.7	7918.5		4134.0	257.0	0.0	0.0
Variable payload and adjustments	*WVPA*	324.7	24.2	7859.9		4134.0	257.0	0.0	0.0
Variable military payload	*WVP*	334.8	23.7	7918.5		4134.0	257.0	0.0	0.0
Total payload and adjustments	*WPA*	1350.4	28.8	38945.5		11577.0	4042.0	802.1	1130.4
Total military payload	*WP*	922.8	33.4	30815.4		11465.0	4042.0	765.0	1089.2

Table C.1. *Payload and adjustments table (continued).*

Appendix D

DDGx Mathcad™ Model

Mathcad™ is a special worksheet, in which all expressions are visualized in their traditional mathematical notation, exactly as they are written on a paper or in class. All entities exist in *regions*; text, math or graphic regions. The execution of the calculations is conducted from top to bottom, and from left to right. The CPU starts at the upper left corner, advances to the right, to the first region, executes it if applicable, and then gets down and proceeds. Refer to [39] for detailed usage instructions. Here only some basic commands are described, to enable understanding of the model. Three kinds of equality signs exist:

$=$	The numerical value of the parameter in the left side is printed.
$:=$	The parameter to the left gets the value of the expression in the right side (can be a numeric value). Only regions that appear after (that is, to the right and lower) this region are influenced by this expression.
$\equiv$	Same as above, but here all expressions, even former ones, are influenced.

Mathcad also tracks the physical units of the arguments, so no unit conversion is needed; this is done internally by the software. At the beginning of the model, Naval Architecture units are inserted, because these units do not exist in Mathcad's unit library, and otherwise are not traceable. Next, all design requirements are input. Numeric data is per Appendix C, which describes the payload of the DDGx ships. Next, all modules are shown. The last page contains the most important data: hull parameters (notice that they use the '≡' sign, so all the document is influenced) and balance status report.

This model contains some conditional expressions. For example, the sonar dome wetted area on page 150, can have three values, as a function of the input. This allows evaluation of different ships. However, it should be remembered that many parameters and expression are problem dependent. The final Fortran model contains only the expression corresponds to the DDGx class.

Gertler residuary drag coefficients are calculated in an external worksheet, as seen in the third page. This file is included in this appendix, right after the model, starting on page 163.

Note that the maximum speed calculation in this model, on page 152, is performed by linear interpolation between the sustained speed and 35 knots. The Fortran self-balanced model conducts the accurate computation, as described in Section 2.5.4.

DDGx Math Model

Units definition

$hp := \frac{33000 \cdot ft \cdot lbf}{min}$ $knt := 1.69 \cdot \frac{ft}{sec}$ $lton \equiv 2240 \cdot lb$ $mile := knt \cdot hr$

Input

Use Appendix C for payload data, and Appendix A for nomenclature.

Trade Off

Average deck height (hull+deckhouse): $H_{DK} := 10.66 \cdot ft$ Raised deck height (if applicable): $H_R := 9.5 \cdot ft$

Sustained (at 80% power) and endurance speeds: $V_S := 30 \cdot knt$ $V_e := 20 \cdot knt$

V_S is set as to balance the resistance and installed propulsion power of the vessel. V_e controls the required fuel amount for the specified range and stores period.

Range and stores period: $E := 3807.6 \cdot mile$ $T_S := 45 \cdot day$

Margins

$KG_{MARG} := 0.5 \cdot ft$ $PMF := 1.08$ $EDMF := 1.0$ $EFMF := 1.01$ $WMF := 0.005$ $E24MF := 1.2$

Physical Parameters

Sea water properties: $\rho_{SW} := 1.9905 \cdot \frac{slug}{ft^3}$ $\upsilon_{SW} := 1.2817 \cdot 10^{-5} \cdot \frac{ft^2}{sec}$

Air properties: $\rho_A := 0.0023817 \cdot \frac{slug}{ft^3}$

Liquids specific volumes: $\gamma_F := 42.3 \cdot \frac{ft^3}{lton}$ $\gamma_{HF} := 43 \cdot \frac{ft^3}{lton}$ $\gamma_{LO} := 39 \cdot \frac{ft^3}{lton}$ $\gamma_W := 36 \cdot \frac{ft^3}{lton}$

Weight Requirements (from Appendix C)

$W_P := 922.8 \cdot lton$ $W_{VP} := 334.8 \cdot lton$ $VCG_P := 33.4 \cdot ft$ $VCG_{VP} := 23.7 \cdot ft$

$W_{P100} := 81.2 \cdot lton$ $W_{P400} := 176.5 \cdot lton$ $W_{P500} := 15.7 \cdot lton$ $W_{P600} := 0 \cdot lton$ $W_7 := 314.6 \cdot lton$

$VCG_{P100} := 40.1 \cdot ft$ $VCG_{P400} := 45.5 \cdot ft$ $VCG_{P500} := 32.2 \cdot ft$ $VCG_{P600} := 0 \cdot ft$ $VCG_{700} := 35.3 \cdot ft$

$W_{F42} := 64.4 \cdot lton$

Sonar dome weights: $W_{498} := 86.5 \cdot lton$ $VCG_{498} := -3.95 \cdot ft$

CPS (30 lton if installed): $W_{CPS} := 30 \cdot lton$

Hull material (will be used for hull structure weight calculation in the weight section later; 1.0 for OS or 0.93 for HTS) and deckhouse material (same use; 1 for aluminum and 2 for steel):

$C_{HMAT} := 0.93$ $C_{DHMAT} := 2$

Area Requirements (from Appendix C)

Required payload deck areas at deckhouse: $A_{DPC} := 3464 \cdot ft^2$ $A_{DPA} := 578 \cdot ft^2$

At hull: $A_{HPC} := 6292 \cdot ft^2$ $A_{HPA} := 5173 \cdot ft^2$

Where 'PC' index stands for C&D (swbs 400) and 'PA' for armament (swbs 500, 600, 700 and F20).

Sonar model (0 for none, 1 for SQS-56, 2 for SQS-53C): $SON_{TYP} := 2$

Manning Requirements

N_O and N_E stand for number of officers and enlisted, respectively:

$N_O := 26$ $N_E := 315$ $N_T := N_E + N_O$ $N_T = 341$ $N_A := 36$

N_T defines the total crew size, N_A the additional accommodations.

Electrical Requirements

$KW_{PAY} := (792.8 + 55) \cdot kW$ (For winter cruise condition, with sonar).

Fin stabilizers: $N_{fins} := 0$ $KW_{fins} := \begin{vmatrix} 0 \cdot kW \text{ if } N_{fins} = 0 \\ 50 \cdot kW \text{ otherwise} \end{vmatrix}$

Machinery

Number of propellers: $N_P := 2$ Type (1 for FP, 2 for CPP): $PROP_{TYP} := 2$

Auxiliary propulsion unit (APU) data: $W_{237} := 0 \cdot lton$ $VCG_{237} := 0 \cdot ft$

Ballast type (1 for compensated, 2 otherwise): $BAL_{TYP} := 1$

Propulsion engines type (1 for diesel/2 for gas turbine/3 for RACER): $PENG_{TYP} := 2$

Data in this model for gas turbine propulsion engines is for the LM2500's; Generator engines for DDA 501-k34's. Number and rated power of these engines (propulsion and generator, respectively):

$N_{PENG} := 4$ $P_{BPENG} := 25775 \cdot hp$ $N_G := 3$ $KW_G := 2500 \cdot kW$

Inlet/exhaust cross section area required for each PE and generator: $A_{IE} := 159.1 \cdot ft^2$ $A_{GIE} := 35.7 \cdot ft^2$

Deckhouse decks impacted by propulsion and generator inlet/exhaust: $N_{DIE} := 2$

Hull decks impacted by propulsion inlet/exhaust: $N_{HPIE} := 2$

A zero actually means that there is no continuous deck in the hull. Hull decks impacted by generator inlet/exhaust:

$N_{HeIE} := 5$

Resistance and Power

Frictional Drag

TSS wetted surface coefficient, assuming no dependency on C_V, according to [21]:

$A_0 := 7.028 - 2.331 \cdot C_{BT} + 0.299 \cdot C_{BT}^2$

$A_1 := -11 + 5.536 \cdot C_{BT} - 0.704 \cdot C_{BT}^2$

$A_2 := 6.913 - 3.419 \cdot C_{BT} + 0.451 \cdot C_{BT}^2$

$C_{STSS} := A_0 + A_1 \cdot C_P + A_2 \cdot C_P^2$ $C_{STSS} = 2.548$

$S_{TSS} := C_{STSS} \cdot \sqrt{V_{FL} \cdot LWL}$ $S_{TSS} = 29307.011 \cdot ft^2$ $S_S := S_{TSS}$

Which means that actual wetted surface area is assumed to be equal to TSS area.

$i := 3..7$ $V_i := i \cdot 5 \cdot knt$ $V_6 := V_S$ $V_4 := V_e$

Correlation allowance: $C_A := 0.0004$

$$S_{SD} := \begin{cases} 0\cdot ft^2 \text{ if } SON_{TYP}=0 \\ 80\cdot ft^2 \text{ if } SON_{TYP}=1 \\ 1400\cdot ft^2 \text{ if } SON_{TYP}=2 \end{cases}$$

ITTC friction expression: $R_{N_i} := LWL\cdot\frac{V_i}{\upsilon_{SW}}$ $C_{F_i} := \frac{0.075}{\left(\log\left(R_{N_i}\right)-2\right)^2}$

$$R_{F_i} := \frac{1}{2}\cdot\rho_{SW}\cdot\left(S_S+S_{SD}\right)\cdot\left(C_{F_i}+C_A\right)\cdot\left(V_i\right)^2$$

Residuary Drag

$C_V := \frac{V_{FL}}{LWL^3}$ $C_V = 0.00282$ $R_i := \frac{V_i}{\sqrt{LWL}}$

Calling the residuary drag coefficients module, which calculates C_R for different beam to draft ratios:

Include:C:\Thesis\Analysis\Residuary_drag_module.MCD

Form factor: $FF := \frac{4}{3}\cdot\left(C_{BT}-3\right)$ $FF = -0.099$

$$C_{RTSS_i} := C_{R3.00_i} + FF\cdot\frac{C_{R3.75_i}-C_{R2.25_i}}{2} + FF^2\cdot\left(\frac{C_{R2.25_i}+C_{R3.75_i}}{2} - C_{R3.00_i}\right)$$

$$R_{RTSS_i} := \frac{1}{2}\cdot\rho_{SW}\cdot\left(S_S+S_{SD}\right)\cdot\left(V_i\right)^2\cdot C_{RTSS_i}$$

Worm curve coefficient: $A := READPRN(Worm\_hull)$

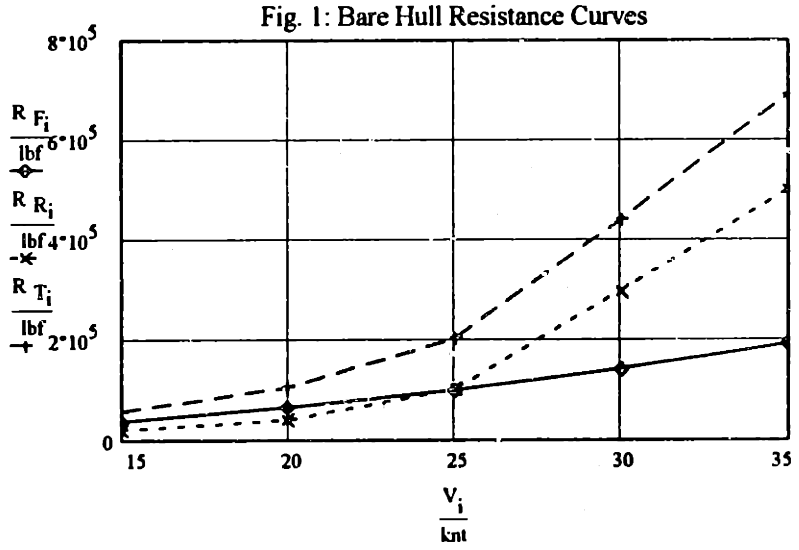
```
WCF_i := | j ← 0
         | $R_i \leftarrow \frac{R_i}{\left(\frac{knt}{\sqrt{ft}}\right)}$
         | while $R_i > A_{j+1,1}$
         |   $j \leftarrow j+1$
         | $A_{j,2} + \frac{A_{j+1,2}-A_{j,2}}{0.05}\cdot\left(R_i - A_{j,1}\right)$
```

$$R_{R_i} := R_{RTSS_i}\cdot WCF_i$$

Bare Hull Resistance

$$R_{T_i} := R_{F_i} + R_{R_i}$$

Fig. 1: Bare Hull Resistance Curves



Ship Effective Horsepower

Bare hull: $P_{EBH_i} := R_{T_i} \cdot V_i$

Appendages drag coefficient, from [18]:

$$C_{DAPP} := \begin{vmatrix} \left(-3\cdot10^{-9}\cdot\frac{LWL^3}{ft^3} + 6\cdot10^{-6}\cdot\frac{LWL^2}{ft^2} - 0.0051\cdot\frac{LWL}{ft} + 3.6432\right)\cdot\frac{hp\cdot10^{-5}}{ft^2\cdot knt^3} \text{ if } 2\cdot N_P + PROP_{TYP}=5 \\ \left(-4\cdot10^{-9}\cdot\frac{LWL^3}{ft^3} + 9\cdot10^{-6}\cdot\frac{LWL^2}{ft^2} - 0.0081\cdot\frac{LWL}{ft} + 5.0717\right)\cdot\frac{hp\cdot10^{-5}}{ft^2\cdot knt^3} \text{ if } 2\cdot N_P + PROP_{TYP}=6 \\ \left(-2\cdot10^{-6}\cdot\frac{LWL^2}{ft^2} - 0.004\cdot\frac{LWL}{ft} + 2.7288\right)\cdot\frac{hp\cdot10^{-5}}{ft^2\cdot knt^3} \text{ if } 2\cdot N_P + PROP_{TYP}=3 \\ \left(-3\cdot10^{-9}\cdot\frac{LWL^3}{ft^3} + 7\cdot10^{-6}\cdot\frac{LWL^2}{ft^2} - 0.0072\cdot\frac{LWL}{ft} + 3.9489\right)\cdot\frac{hp\cdot10^{-5}}{ft^2\cdot knt^3} \text{ if } 2\cdot N_P + PROP_{TYP}=4 \end{vmatrix}$$

$$C_{DAPP} = 2.848\cdot\frac{hp\cdot10^{-5}}{ft^2\cdot knt^3}$$

Estimated propeller size, from [18] and [23]:

$C_{PROP} := if(N_P>1,1,1.2)$ $C_{PROP} = 1$ $D_P := (0.64\cdot T + 0.013\cdot LWL)\cdot C_{PROP}$ $D_P = 19.144\cdot ft$

Roll fins: $P_{Efins_i} := \begin{vmatrix} 0 \text{ if } N_{fins}=0 \\ 0.025\cdot P_{EBH_i} \text{ otherwise} \end{vmatrix}$

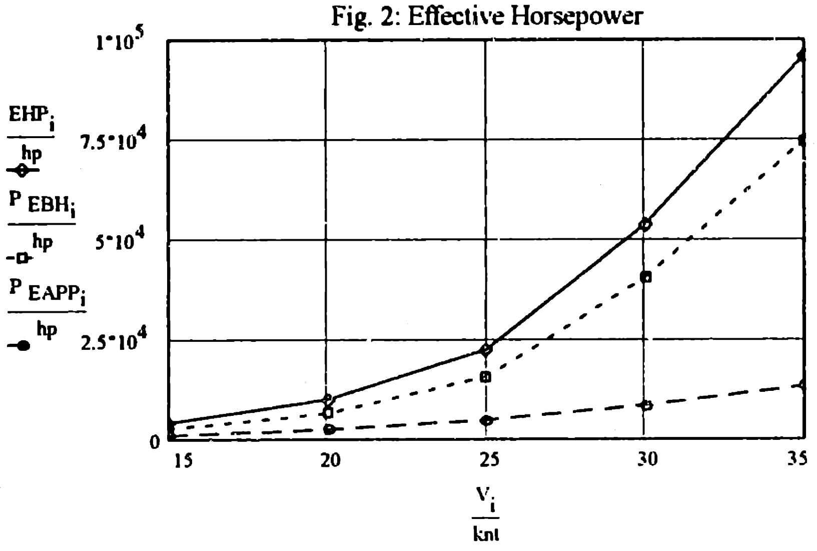
$P_{EAPP_i} := 1.23\cdot LWL\cdot D_P\cdot C_{DAPP}\cdot (V_i)^3 + P_{Efins_i}$ (Scaled to fit V_s=30 knots for DDG51).

Air frontal area (+5% for masts, equip., etc): $A_W := 1.05\cdot B\cdot(D_{10} - T + 3\cdot H_{DK})$ $A_W = 3352.058\cdot ft^2$

$C_{AA} := 0.7$ $\quad P_{EAA_i} := \frac{1}{2} \cdot C_{AA} \cdot A_W \cdot \rho_A \cdot (V_i)^3$

Total effective horsepower: $P_{ET_i} := P_{EBH_i} + P_{EAPP_i} + P_{EAA_i}$ $\quad EHP_i := PMF \cdot P_{ET_i}$

Fig. 2: Effective Horsepower



<u>Power Balance</u>

Approximated propulsive coefficient: $PC := \mathrm{if}(PROP_{TYP} = 1, 0.7, 0.67)$ $\quad PC = 0.67$

$SHP_i := \frac{EHP_i}{PC}$ $\quad SHP_S := SHP_6$ $\quad SHP_S = 80000.714 \cdot hp$ $\quad SHP_e := SHP_4$ $\quad SHP_e = 14935.726 \cdot hp$

Transmission efficiency: $\eta := 0.97$

Required installed power (with 25% for fouling and sea state): $P_{IREQ} := \frac{1.25 \cdot SHP_S}{\eta}$ $\quad P_{IREQ} = 103093.7 \cdot hp$

Actual installed power: $P_I := N_{PENG} \cdot P_{BPENG}$ $\quad P_I = 103100 \cdot hp$ $\quad ERR_V := \frac{P_I - P_{IREQ}}{P_I}$

Maximal continuous brake horsepower: $P_{BMAX} := N_{PENG} \cdot P_{BPENG}$ $\quad P_{BMAX} = 103100 \cdot hp$

$V_{MAX} := V_S + \frac{\eta \cdot P_{BMAX} - SHP_S}{SHP_7 - SHP_S} \cdot (V_7 - V_S)$ $\quad V_{MAX} = 31.581 \cdot knt$

<u>Space Available</u>

Underwater available hull volume: $V_{HUW} := V_{FL}$

$H_{MBMIN} := 22 \cdot ft$

For sheer line, 3 criteria exist: - deck edge above water at 25^0 heel (DDS-079-1).
- ensure longitudinal strength.
- contain machinery box (in height).

$M := \begin{bmatrix} 0.21 \cdot B + T \\ \dfrac{LWL}{15} \\ H_{MBMIN} + H_{DK} \end{bmatrix}$ $M = \begin{pmatrix} 33.033 \\ 31.019 \\ 32.66 \end{pmatrix} \cdot ft$ $D_{10MIN} := \max(M)$ $D_{10MIN} = 33.033 \cdot ft$

$D_{0MIN} := 2.011827 \cdot T - \dfrac{6.36215 \cdot 10^{-6}}{ft} \cdot LWL^2 + 2.780649 \cdot 10^{-2} \cdot LWL$ $D_{0MIN} = 52.725 \cdot ft$ $D_0 := D_{0MIN}$

$D_{20MIN} := 0.014 \cdot LWL \cdot \left(2.125 + \dfrac{1.25 \cdot 10^{-3}}{ft} \cdot LWL\right) + T$ $D_{20MIN} = 38.092 \cdot ft$ $D_{20} := D_{20MIN}$

$F_0 := D_0 - T$ $F_{10} := D_{10} - T$ $F_{20} := D_{20} - T$ (Assuming zero trim).

$A_{PRO} := \dfrac{LWL}{0.98} \cdot \dfrac{F_0 + 4 \cdot F_{10} + F_{20}}{6}$ (Trapezoidal rule).

$F_{AV} := \dfrac{A_{PRO}}{LWL}$ $F_{AV} = 23.005 \cdot ft$ $D_{AV} := F_{AV} + T$ $D_{AV} = 43.466 \cdot ft$

Cubic number: $CN := \dfrac{LWL \cdot B \cdot D_{AV}}{10^5 \cdot ft^3}$ $CN = 12.108$

Water plane coefficient: $C_W := 0.278 + 0.836 \cdot C_P$ $C_W = 0.787$

Above water hull volume: $V_{HAW} := F_{AV}^2 \cdot \tan(10 \cdot deg) \cdot LWL + LWL \cdot B \cdot C_W \cdot F_{AV}$

$V_{HAW} = 547836.229 \cdot ft^3$

Lost hull volume due to raised deck: $V_{HL} := \left| \begin{array}{l} B_{max} \leftarrow 2 \cdot F_{AV} \cdot \tan(10 \cdot deg) + B \\ B_{low} \leftarrow B_{max} - 2 \cdot H_R \cdot \tan(10 \cdot deg) \\ C_{RD} \cdot LWL \cdot \left(\dfrac{B_{max} + B_{low}}{2}\right) \cdot H_R \cdot C_W \end{array} \right.$

Total hull volume: $V_{HT} := V_{HUW} + V_{HAW} - V_{HL}$ $V_{HT} = 785978.989 \cdot ft^3$

Deckhouse size upper limit: $V_{DMAX} := \left(2 \cdot B - 4 \cdot \dfrac{H_{DK}}{\tan(80 \cdot deg)}\right) \cdot H_{DK} \cdot 0.6 \cdot LWL \cdot 1.1$

Total ship volume: $V_T := V_{HT} + V_D$ $V_T = 977987.989 \cdot ft^3$

Electrical Load

Estimated maximum functional loads for winter cruise condition:

$KW_P := 0.00323 \cdot \dfrac{kW}{hp} \cdot P_I$ (SWBS 200). $KW_P = 333.013 \cdot kW$

$KW_S := 0.00826 \cdot \dfrac{kW}{ft^2} \cdot LWL \cdot T$ (SWBS 561). $KW_S = 78.637 \cdot kW$

$KW_E := 0.000213 \cdot \frac{kW}{ft^3} \cdot V_T$ **(SWBS 300).** $KW_E = 208.311 \cdot kW$

$KW_M := 101.4 \cdot kW$ **(SWBS 430+475).**

$KW_{CPS} := \begin{cases} 0 \cdot kW & \text{if } W_{CPS} = 0 \cdot lton \\ 0.000135 \cdot \frac{kW}{ft^3} \cdot V_T & \text{otherwise} \end{cases}$ $KW_{CPS} = 132.028 \cdot kW$

$KW_B := 0.235 \cdot N_T \cdot kW$ **(SWBS 517).** $KW_B = 80.135 \cdot kW$

$KW_F := 0.000097 \cdot \frac{kW}{ft^3} \cdot V_T$ **(SWBS 521).** $KW_F = 94.865 \cdot kW$

$KW_{HN} := 0.000177 \cdot \frac{kW}{ft^3} \cdot V_{HT}$ **(SWBS 540).** $KW_{HN} = 139.118 \cdot kW$

$KW_A := 0.65 \cdot N_T \cdot kW + KW_{fins}$ **(SWBS 530+550).** $KW_A = 221.65 \cdot kW$

$KW_{SERV} := 0.395 \cdot N_T \cdot kW$ **(SWBS 600).** $KW_{SERV} = 134.695 \cdot kW$

The calculation is iterative, because KW_H, KW_V and KW_{AC} depend on V_{AUX}, which depends on the maximum functional load. Non-payload functional load (without the above mentioned loads):

$$KW_{NP} := KW_P + KW_S + KW_E + KW_M + KW_B + KW_F + KW_{HN} + KW_A + KW_{SERV}$$

```
KW_MFL :=
    $KW_{MFL} \leftarrow 1000 \cdot kW$
    $KW_X \leftarrow 0 \cdot kW$
    while $\left| \frac{KW_{MFL} - KW_X}{KW_{MFL}} \right| > 0.01$
        $KW_X \leftarrow KW_{MFL}$
        $V_{AUX} \leftarrow 56900 \cdot \frac{ft^3}{kW} \cdot \frac{KW_X}{3411}$
        $KW_H \leftarrow 0.00064 \cdot \frac{kW}{ft^3} \cdot (V_T - V_{MB} - V_{AUX})$
        $KW_V \leftarrow 0.103 \cdot (KW_H + KW_{PAY}) + KW_{CPS}$
        $KW_{AC} \leftarrow 0.67 \cdot \left[ 0.1 \cdot kW \cdot N_T + 0.00067 \cdot \frac{kW}{ft^3} \cdot (V_T - V_{MB} - V_{AUX}) + 0.1 \cdot KW_{PAY} \right]$
        $KW_{MFL} \leftarrow KW_{NP} + KW_H + KW_V + KW_{AC} + KW_{PAY}$
    $KW_{MFL}$
```

$KW_{MFL} = 3441.663 \cdot kW$

$KW_{MFLM} := EDMF \cdot EFMF \cdot KW_{MFL}$ $KW_{MFLM} = 3476.08 \cdot kW$

The iterative process yields:

$V_{AUX} := 56900 \cdot \frac{ft^3}{kW} \cdot \frac{KW_{MFL}}{3411}$

$KW_H := 0.00064 \cdot \frac{kW}{ft^3} \cdot (V_T - V_{MB} - V_{AUX})$ $KW_H = 500.452 \cdot kW$

$KW_{AC} := 0.67 \cdot \left[0.1 \cdot kW \cdot N_T + 0.00067 \cdot \frac{kW}{ft^3} \cdot (V_T - V_{MB} - V_{AUX}) + 0.1 \cdot KW_{PAY}\right]$ $KW_{AC} = 430.67 \cdot kW$

$KW_V := 0.103 \cdot (KW_H + KW_{PAY}) + KW_{CPS}$ $KW_V = 270.898 \cdot kW$

Power required per generator, with one in stand-by position:

$KW_{GREQ} := \frac{KW_{MFLM}}{(N_G - 1) \cdot 0.9}$ $KW_{GREQ} = 1931.155 \cdot kW$

The 0.9 compensates for possible voltage fluctuations. 24 hours electrical load:

$KW_{24} := 0.5 \cdot (KW_{MFL} - KW_P - KW_S) + 1 \cdot (KW_P + KW_S)$ $KW_{24} = 1926.657 \cdot kW$

Including design margin: $KW_{24AVG} := E24MF \cdot KW_{24}$ $KW_{24AVG} = 2311.988 \cdot kW$

Space requirements

Machinery box

Since the same HM&E system is used, the same propulsion system in particular, It is assumed that the volume of main machinery rooms (MMR's) is constant. Likewise, it is further assumed that the volume of all other machinery rooms (auxiliary and others) is proportional to the electric load the ship requires. That is:

$V_{MB} \equiv 138620 \cdot ft^3$ $H_{MB} := D_{10} - \frac{N_{HPIE}}{2} \cdot H_{DK}$ $H_{MB} = 31.145 \cdot ft$

$V_{AUX} := 56900 \cdot \frac{ft^3}{kW} \cdot \frac{KW_{MFL}}{3411}$ $V_{AUX} = 57411.498 \cdot ft^3$ (3411 kW taken from the match run).

Tankage

Fuel

Based on [29].
Average endurance brake horsepower required (including 10% margin for fouling and sea state):

$P_{eBAVG} := \frac{1.1 \cdot SHP_e}{\eta}$ $P_{eBAVG} = 16937.422 \cdot hp$

$r_{pPE} := \frac{P_{eBAVG}}{\left(\frac{P_I}{2}\right)}$ $r_{nPE} := \frac{V_e}{V_{MAX}}$ $r_{QPE} := \frac{r_{pPE}}{r_{nPE}}$ $SFC_{PE} := 0.4097 \cdot \frac{lb}{hp \cdot hr}$

$SFC_{ePE} := \frac{SFC_{PE}}{r_{pPE}} \cdot \left(7.215 \cdot 10^{-2} \cdot e^{1.252 \cdot r_{nPE}} + 0.3629 \cdot r_{QPE} \cdot e^{0.7229 \cdot r_{nPE}}\right)$ $SFC_{ePE} = 0.5699 \cdot \frac{lb}{hp \cdot hr}$

Correction for instrumentation inaccuracy and machinery design changes:

$$f_l := \begin{cases} 1.04 & \text{if } 1.1 \cdot SHP_e \le \frac{1}{3} \cdot \frac{P_I}{2} \\ 1.03 & \text{if } 1.1 \cdot SHP_e \ge \frac{2}{3} \cdot \frac{P_I}{2} \\ 1.02 & \text{otherwise} \end{cases}$$ $f_l = 1.04$

Specified fuel rate: $FR_{SP} := f_l \cdot SFC_{ePE}$

Average fuel rate allowing for plant deterioration: $FR_{AVG} := 1.05 \cdot FR_{SP}$ $FR_{AVG} = 0.622 \cdot \frac{lb}{hp \cdot hr}$

Burnable propulsion endurance fuel weight: $W_{BP} := \frac{E}{V_e} \cdot P_{eBAVG} \cdot FR_{AVG}$ $W_{BP} = 895.847 \cdot lton$

Tailpipe allowance: $TPA := 0.95$

Required propulsion fuel weight: $W_{FP} := \frac{W_{BP}}{TPA}$ $W_{FP} = 942.997 \cdot lton$

Required propulsion fuel tank volume (including allowance for expansion and tank internal structure):

$V_{FP} := 1.02 \cdot 1.05 \cdot \gamma_F \cdot W_{FP}$ $V_{FP} = 42720.868 \cdot ft^3$

$P_{G24} := \frac{KW_{24AVG}}{0.7457 \cdot 0.961 \cdot 0.989} \cdot \frac{hp}{kW}$ $r_{pG} := \frac{P_{G24}}{2 \cdot 4600 \cdot hp}$ $r_{nG} := 1$ $r_{QG} := \frac{r_{pG}}{r_{nG}}$ $SFC_G := 0.4727 \cdot \frac{lb}{hp \cdot hr}$

$SFC_{G24} := \frac{SFC_G}{r_{pG}} \cdot \left(0.2821 + 0.7179 \cdot r_{QG}\right)$ $SFC_{G24} = 0.715 \cdot \frac{lb}{hp \cdot hr}$

$SFC_{GE24} := \frac{SFC_{G24} \cdot P_{G24}}{KW_{24AVG}}$ $SFC_{GE24} = 1.009 \cdot \frac{lb}{kW \cdot hr}$

Margin for instrumentation inaccuracy and machinery design changes: $f_{le} := 1.04$

Specified fuel rate: $FR_{GSP} := f_{le} \cdot SFC_{GE24}$

Average fuel rate, allowing for plant deterioration: $FR_{GAVG} := 1.05 \cdot FR_{GSP}$ $FR_{GAVG} = 1.102 \cdot \frac{lb}{kW \cdot hr}$

Burnable electrical endurance fuel weight:

$W_{Be} := \frac{E}{V_e} \cdot KW_{24AVG} \cdot FR_{GAVG}$ $W_{Be} = 216.602 \cdot lton$

Required electrical fuel weight: $W_{Fe} := \frac{W_{Be}}{TPA}$ $W_{Fe} = 228.002 \cdot lton$

Required electrical fuel volume: $V_{Fe} := 1.02 \cdot 1.05 \cdot \gamma_F \cdot W_{Fe}$ $V_{Fe} = 10329.255 \cdot ft^3$

Total fuel weight and tanks volume: $W_{F41} := W_{FP} + W_{Fe}$ $W_{F41} = 1170.999 \cdot lton$

$V_F := V_{FP} + V_{Fe}$ $\qquad V_F = 53050.123 \cdot ft^3$

<u>Other Tanks</u>

Helicopter fuel: $V_{HF} := 1.02 \cdot 1.05 \cdot W_{F42} \cdot \gamma_{HF}$ $\qquad V_{HF} = 2965.813 \cdot ft^3$

Lubrication oil: $W_{F46} := 17.6 \cdot lton$ $\qquad V_{LO} := 1.02 \cdot 1.05 \cdot W_{F46} \cdot \gamma_{LO}$ $\qquad V_{LO} = 735.134 \cdot ft^3$

Potable water: $W_{F52} := N_T \cdot 0.15 \cdot lton$ $\qquad W_{F52} = 51.15 \cdot lton$

$V_W := 1.02 \cdot W_{F52} \cdot \gamma_W$ $\qquad V_W = 1878.228 \cdot ft^3$ (Water does not expand).

Sewage: $V_{SEW} := (N_T + N_A) \cdot 2.005 \cdot ft^3$ $\qquad V_{SEW} = 755.885 \cdot ft^3$

Waste oil: $V_{WASTE} := 0.02 \cdot V_F$ $\qquad V_{WASTE} = 1061.002 \cdot ft^3$

Clean ballast: $V_{BAL} := 0.19 \cdot V_F$ $\qquad V_{BAL} = 10079.523 \cdot ft^3$

Total tankage volume required:

$V_{TK} := V_F + V_{HF} + V_{LO} + V_W + V_{SEW} + V_{WASTE} + V_{BAL}$ $\qquad V_{TK} = 70525.709 \cdot ft^3$

<u>Payload Deck Areas</u>

Deckhouse payload area (including access): $A_{DPR} := 1.15 \cdot A_{DPA} + 1.23 \cdot A_{DPC}$ $\qquad A_{DPR} = 4925.42 \cdot ft^2$

Hull payload area (including access): $A_{HPR} := 1.15 \cdot A_{HPA} + 1.23 \cdot A_{HPC}$ $\qquad A_{HPR} = 13688.11 \cdot ft^2$

<u>Living Deck Area</u>

The assumption is that officers live at deckhouse, and enlisted at hull. At deckhouse:

$A_{COXO} := 225 \cdot ft^2$ $\qquad A_{DO} := 75 \cdot N_O \cdot ft^2$ $\qquad A_{DL} := A_{COXO} + A_{DO}$ $\qquad A_{DL} = 2175 \cdot ft^2$

At hull: $A_{HAB} := 50 \cdot ft^2$ $\qquad A_{HL} := A_{HAB} \cdot (N_T + N_A) - A_{DL}$ $\qquad A_{HL} = 16675 \cdot ft^2$

<u>Other Ship areas</u>

Hull stores: $A_{HS} := 300 \cdot ft^2 + 0.0158 \cdot \frac{ft^2}{lb} \cdot N_T \cdot 9 \cdot \frac{lb}{day} \cdot T_S$ $\qquad A_{HS} = 2482.059 \cdot ft^2$

Deckhouse maintenance: $A_{DM} := 0.05 \cdot (A_{DPR} + A_{DL})$ $\qquad A_{DM} = 355.021 \cdot ft^2$

Bridge and chart rooms: $A_{DB} := 16 \cdot ft \cdot (B - 18 \cdot ft)$ $\qquad A_{DB} = 669.901 \cdot ft^2$

Ship functions at hull: $A_{HSF} := 1750 \cdot ft^2 \cdot CN$ $\qquad A_{HSF} = 21189.012 \cdot ft^2$

Total inlet/exhaust cross section area needed: $A_{PIE} := N_{PENG} \cdot A_{IE}$ $\qquad A_{PIE} = 636.4 \cdot ft^2$

$A_{eIE} := N_G \cdot A_{GIE}$ $\qquad A_{eIE} = 107.1 \cdot ft^2$

Engine inlet/exhaust area at deckhouse: $A_{DIE} := 1.4 \cdot N_{DIE} \cdot (A_{PIE} + A_{eIE})$ $\qquad A_{DIE} = 2081.8 \cdot ft^2$

Engine inlet/exhaust area at hull: $A_{HIE} := 1.4 \cdot (N_{HPIE} \cdot A_{PIE} + N_{HeIE} \cdot A_{eIE})$ $\qquad A_{HIE} = 2531.62 \cdot ft^2$

<u>Total Required Area/Volume</u>

Hull: $A_{HR} := A_{HPR} + A_{HL} + A_{HS} + A_{HSF} + A_{HIE}$ $A_{HR} = 56565.801 \cdot ft^2$

$V_{HR} := H_{DK} \cdot A_{HR}$ $V_{HR} = 602991.438 \cdot ft^3$

Deckhouse: $A_{DR} := A_{DPR} + A_{DL} + A_{DM} + A_{DB} + A_{DIE}$ $A_{DR} = 10207.142 \cdot ft^2$

$V_{DR} := H_{DK} \cdot A_{DR}$ $V_{DR} = 108808.139 \cdot ft^3$

Total: $A_{TR} := A_{HR} + A_{DR}$ $A_{TR} = 66772.943 \cdot ft^2$

$V_{TR} := V_{HR} + V_{DR}$ $V_{TR} = 711799.576 \cdot ft^3$

Available hull volume: $V_{HA} := V_{HT} - V_{MB} - V_{AUX} - V_{TK}$ $V_{HA} = 519421.781 \cdot ft^3$

Available hull area: $A_{HA} := \frac{V_{HA}}{H_{DK}}$ $A_{HA} = 48726.246 \cdot ft^2$

Total available volume: $V_{TA} := V_{HA} + V_D$ $V_{TA} = 711430.781 \cdot ft^3$

Available deckhouse area: $A_{DA} := \frac{V_D}{H_{DK}}$ $A_{DA} = 18012.101 \cdot ft^2$

Total available area: $A_{TA} := A_{HA} + A_{DA}$ $A_{TA} = 66738.347 \cdot ft^2$

Area effectiveness: $ERR_A := \frac{A_{TA} - A_{TR}}{A_{TR}}$ $ERR_A = -0.001$

<u>Weight</u>

<u>SWBS 200</u>

Basic machinery (239+241/242+250 .. 290):

$W_{BM} := P_I \cdot \frac{lb}{hp} \cdot \left[9.0 + 12.4 \cdot \left(P_I \cdot \frac{10^{-5}}{hp} - 1 \right)^2 \right]$ $W_{BM} = 414.79 \cdot lton$

Shafting (SWBS 243): $f_S := if(N_P = 1, 0.33, 0.5)$ $W_S := 0.82 \cdot \frac{lton}{ft} \cdot LWL \cdot f_S$ $W_S = 190.768 \cdot lton$

Propulsors (245): $W_{PR} := 0.087 \cdot lb \cdot \left(\frac{D_P}{ft} \right)^{5.497 - \frac{0.0433}{ft} \cdot D_P} \cdot N_P$ $W_{PR} = 74.972 \cdot lton$

Bearings (244): $W_B := 0.235 \cdot (W_S + W_{PR})$ $W_B = 62.449 \cdot lton$

Total shafting: $W_{ST} := W_S + W_B + W_{PR}$ $W_{ST} = 328.189 \cdot lton$

Total for propulsion: $W_2 := W_{BM} + W_{ST} + W_{237}$ $W_2 = 742.979 \cdot lton$

<u>SWBS 300</u>

$W_3 := 50 \cdot lton + 0.0352 \cdot \frac{lton}{kW} \cdot N_G \cdot KW_G$ $W_3 = 314 \cdot lton$

SWBS 400

Gyro/IC/Navigation (420+430): $W_{IC} := 4.45 \cdot 10^{-5} \cdot \frac{\text{lton}}{\text{ft}^3} \cdot V_T$ $\quad W_{IC} = 43.52 \cdot \text{lton}$

Other/Misc: $W_{CO} := 2.2 \cdot CN \cdot \text{lton}$ $\quad W_{CO} = 26.638 \cdot \text{lton}$

Cabling: $W_{CC} := 0.4 \cdot (W_{P400} + W_{IC} + W_{CO})$ $\quad W_{CC} = 98.663 \cdot \text{lton}$

Total: $W_4 := W_{P400} + W_{IC} + W_{CO} + W_{CC} + W_{498}$ $\quad W_4 = 431.821 \cdot \text{lton}$

SWBS 500

Aux systems operating fluids: $W_{598} := 0.000062 \cdot V_T \cdot \frac{\text{lton}}{\text{ft}^3}$ $\quad W_{598} = 60.635 \cdot \text{lton}$

$$W_{AUX} := \left[0.000772 \cdot \left(\frac{V_T}{\text{ft}^3}\right)^{1.443} + 5.14 \cdot \frac{V_T}{\text{ft}^3} + 6.19 \cdot \left(\frac{V_T}{\text{ft}^3}\right)^{0.7224} + 377 \cdot N_T + 2.74 \cdot \frac{P_I}{\text{hp}}\right] \cdot 10^{-4} \cdot \text{lton} + 117 \cdot \text{lton}$$

Environmental support: $W_{593} := 10 \cdot \text{lton}$

Total: $W_5 := W_{AUX} + W_{P500} + W_{593} + W_{598} + W_{CPS}$ $\quad W_5 = 824.297 \cdot \text{lton}$

SWBS 600

Hull fittings (610+620+630): $W_{OFH} := \frac{0.000418}{\text{ft}^3} \cdot V_T \cdot \text{lton}$ $\quad W_{OFH} = 408.799 \cdot \text{lton}$

Personnel related (640+650+660+670): $W_{OFP} := 0.8 \cdot (N_T - 9.5) \cdot \text{lton}$ $\quad W_{OFP} = 265.2 \cdot \text{lton}$

Total: $W_6 := W_{OFH} + W_{OFP} + W_{P600}$ $\quad W_6 = 673.999 \cdot \text{lton}$

SWBS 100

Hull (110 .. 140,160,190):

$W_{BH} := C_{HMAT} \cdot 1.135 \cdot (1.68341 \cdot CN^2 + 167.1721 \cdot CN - 103.283) \cdot \text{lton}$ $\quad W_{BH} = 2288.044 \cdot \text{lton}$

Deckhouse (150): $\rho_{DH} := \text{if}(C_{DHMAT} = 1, 0.000746 \cdot 1.02, 0.00168)$

$W_{DH} := \rho_{DH} \cdot \frac{\text{lton}}{\text{ft}^3} \cdot V_D$ $\quad W_{DH} = 322.575 \cdot \text{lton}$

Masts (171): $W_{171} := 2 \cdot \text{lton}$ (Assuming the same mast).

Foundations (180): $W_{180} := 0.0735 \cdot (W_{BH} + W_2 + W_3 + W_4 + W_5 + W_6 + W_7)$ $\quad W_{180} = 410.846 \cdot \text{lton}$

Total: $W_1 := W_{BH} + W_{DH} + W_{171} + W_{180} + W_{P100}$ $\quad W_1 = 3104.665 \cdot \text{lton}$

Weight Summary

Margin for future growth: $W_{M24} := WMF \cdot \sum_{i=1}^{7} W_i$ $\quad W_{M24} = 32.032 \cdot \text{lton}$

Lightship weight: $W_{LS} := \sum_{i=1}^{7} W_i + W_{M24}$ $W_{LS} = 6438.393 \cdot \text{lton}$

Provisions: $W_{F31} := N_T \cdot 2.45 \cdot 10^{-3} \cdot \frac{\text{lton}}{\text{day}} \cdot T_S$ $W_{F31} = 37.595 \cdot \text{lton}$

General stores: $W_{F32} := 0.00071 \cdot \frac{\text{lton}}{\text{day}} \cdot T_S \cdot N_T + 0.0049 \cdot \text{lton} \cdot N_T$ $W_{F32} = 12.566 \cdot \text{lton}$

Crew: $W_{F10} := 236 \cdot \text{lb} \cdot N_E + 400 \cdot \text{lb} \cdot (N_O + 1)$ $W_{F10} = 38.009 \cdot \text{lton}$

Total weight:

$W_T := W_{LS} + W_{VP} + W_{F41} + W_{F46} + W_{F52} + W_{F31} + W_{F32} + W_{F10}$ $W_T = 8101.112 \cdot \text{lton}$

$ERR := \frac{W_{FL} - W_T}{W_T}$ $ERR = 0.003$ $F_P := \frac{W_P}{W_T}$ $F_P = 0.114$

Lightship weight groups center of gravity and moments:

$VCG_{BH} := 0.51 \cdot D_{10}$ $VCG_{BH} = 21.32 \cdot \text{ft}$ $P_1 := W_{BH} \cdot VCG_{BH}$

$VCG_{DH} := D_{10} + 1.5 \cdot H_{DK}$ $VCG_{DH} = 57.795 \cdot \text{ft}$ $P_2 := W_{DH} \cdot VCG_{DH}$

$VCG_{180} := 0.5 \cdot D_{10}$ $VCG_{180} = 20.902 \cdot \text{ft}$ $P_3 := W_{180} \cdot VCG_{180}$

$VCG_{171} := D_{10} + 0.13 \cdot LWL$ $VCG_{171} = 102.292 \cdot \text{ft}$ $P_4 := W_{171} \cdot VCG_{171}$

$P_{100} := P_1 + P_2 + P_3 + P_4 + W_{P100} \cdot VCG_{P100}$ $VCG_{100} := \frac{P_{100}}{W_1}$ $VCG_{100} = 25.598 \cdot \text{ft}$

$VCG_{BM} := 0.5 \cdot D_{10}$ $VCG_{BM} = 20.902 \cdot \text{ft}$ $P_5 := W_{BM} \cdot VCG_{BM}$

$VCG_{ST} := 4.8 \cdot \text{ft} + 0.35 \cdot T$ $VCG_{ST} = 11.961 \cdot \text{ft}$ $P_6 := W_{ST} \cdot VCG_{ST}$

$P_7 := W_{237} \cdot VCG_{237}$

$P_{200} := P_5 + P_6 + P_7$ $VCG_{200} := \frac{P_{200}}{W_2}$ $VCG_{200} = 16.953 \cdot \text{ft}$

$VCG_{300} := 0.55 \cdot D_{10}$ $VCG_{300} = 22.993 \cdot \text{ft}$ $P_{300} := W_3 \cdot VCG_{300}$

$VCG_{IC} := D_{10}$ $VCG_{IC} = 41.805 \cdot \text{ft}$ $P_9 := W_{IC} \cdot VCG_{IC}$

$VCG_{CO} := 5.6 \cdot \text{ft} + 0.4625 \cdot D_{10}$ $VCG_{CO} = 24.935 \cdot \text{ft}$ $P_{10} := W_{CO} \cdot VCG_{CO}$

$VCG_{CC} := 0.5 \cdot D_{10}$ $VCG_{CC} = 20.902 \cdot \text{ft}$ $P_{11} := W_{CC} \cdot VCG_{CC}$

$P_{12} := W_{498} \cdot VCG_{498}$

$P_{400} := P_9 + P_{10} + P_{11} + P_{12} + W_{P400} \cdot VCG_{P400}$ $VCG_{400} := \frac{P_{400}}{W_4}$ $VCG_{400} = 28.333 \cdot \text{ft}$

$VCG_{AUX} := 0.9 \cdot (D_{10} - 9.4 \cdot \text{ft})$ $VCG_{AUX} = 29.164 \cdot \text{ft}$ $P_{13} := W_{AUX} \cdot VCG_{AUX}$

$P_{500} := P_{13} + W_{P500} \cdot VCG_{P500}$ $VCG_{500} := \frac{P_{500}}{W_5}$ $VCG_{500} = 25.662 \cdot \text{ft}$

$VCG_{OFH} := 0.65 \cdot D_{10}$ $VCG_{OFH} = 27.173 \cdot \text{ft}$ $P_{15} := W_{OFH} \cdot VCG_{OFH}$

OFH 10 OFH 15 OFH OFH

$VCG_{OFP} := 4.2\cdot ft + 0.4\cdot D_{10}$ $VCG_{OFP} = 20.922\cdot ft$ $P_{16} := W_{OFP}\cdot VCG_{OFP}$

$P_{600} := P_{15} + P_{16} + W_{P600}\cdot VCG_{P600}$ $VCG_{600} := \frac{P_{600}}{W_6}$ $VCG_{600} = 24.713\cdot ft$

Total lightship vertical moment is (note that variable payload is deducted):

$P_{WG} := P_{100} + P_{200} + P_{300} + P_{400} + P_{500} + P_{600} + W_7\cdot VCG_{700}$ $P_{WG} = 160438.982\cdot lton\cdot ft$

Vertical CG of lightship: $VCG_{LS} := \frac{P_{WG}}{W_{LS} - W_{M24}}$ $VCG_{LS} = 25.044\cdot ft$

Here we assume that the weight margin's CG location is at the CG of the lightship.

$KG_{LS} := VCG_{LS}$

Variable loads weight group center of gravity and moments:

$VCG_{F10} := 0.732\cdot D_{10}$	$VCG_{F10} = 30.601\cdot ft$	$P_{17} := W_{F10}\cdot VCG_{F10}$
$VCG_{F31} := 0.523\cdot D_{AV}$	$VCG_{F31} = 22.733\cdot ft$	$P_{18} := W_{F31}\cdot VCG_{F31}$
$VCG_{F32} := 0.592\cdot D_{AV}$	$VCG_{F32} = 25.732\cdot ft$	$P_{19} := W_{F32}\cdot VCG_{F32}$
	$VCG_{F41} := 10.3\cdot ft$	$P_{20} := W_{F41}\cdot VCG_{F41}$
$VCG_{F46} := 0.53\cdot H_{MB}$	$VCG_{F46} = 16.507\cdot ft$	$P_{21} := W_{F46}\cdot VCG_{F46}$
$VCG_{F52} := 0.138\cdot D_{AV}$	$VCG_{F52} = 5.998\cdot ft$	$P_{22} := W_{F52}\cdot VCG_{F52}$

Total moment: $P_{WGL} := P_{17} + P_{18} + P_{19} + P_{20} + P_{21} + P_{22} + W_{VP}\cdot VCG_{VP}$

Total variable loads weight: $W_L := W_{F10} + W_{F31} + W_{F32} + W_{F41} + W_{F46} + W_{F52} + W_{VP}$

Vertical center of gravity: $VCG_L := \frac{P_{WGL}}{W_L}$ $VCG_L = 13.793\cdot ft$

$KG := \frac{W_{LS}\cdot KG_{LS} + W_L\cdot VCG_L}{W_T} + KG_{MARG}$ $KG = 23.235\cdot ft$

Stability

$C_{IT} := -0.537 + 1.44\cdot C_W$ $C_{IT} = 0.596$

$KB := \frac{T}{3}\cdot\left(2.4 - \frac{C_P\cdot C_X}{C_W}\right)$ $KB = 12.047\cdot ft$ $BM := \frac{LWL\cdot B^3\cdot C_{IT}}{12\cdot V_{FL}}$ $BM = 17.457\cdot ft$

$GM := KB + BM - KG$ $GM = 6.27\cdot ft$ $C_{GMB} := \frac{GM}{B}$ $C_{GMB} = 0.105$

Design Parameters and Summary

Hull characteristics

$C_P \equiv 0.609$ **(0.50-0.70 allowed).** $C_X \equiv 0.819$ **(0.70-0.90 allowed).** $C_{RD} \equiv 0.2$

$C_{\Delta L} \equiv 80.68 \cdot \frac{\text{lton}}{\text{ft}^3}$ **(60-90 allowed).** $C_{BT} \equiv 2.926$ **(2.8-3.7 allowed).** $C_{D10} \equiv 11.13$ **(10-15 allowed).**

Hull dimensions

Note that only the full load displacement is to be specified; all other dimensions are evaluated using the above hull characteristics. After weight convergence check for other requirements compliance.

$W_{FL1} := \frac{W_P}{0.1}$ $W_{FL1} = 9228 \cdot \text{lton}$ **(First iteration only).**

$W_T = 8101.112 \cdot \text{lton}$ $W_{FL} \equiv 8127 \cdot \text{lton}$ $V_{FL} \equiv W_{FL} \cdot 34.98 \cdot \frac{\text{ft}^3}{\text{lton}}$ $F_P = 0.1139$

$LWL \equiv 100 \cdot \left(\frac{W_{FL}}{C_{\Delta L}}\right)^{\frac{1}{3}}$ $LWL = 465.288 \cdot \text{ft}$ $B \equiv \sqrt{\frac{C_{BT} \cdot V_{FL}}{C_P \cdot C_X \cdot LWL}}$ $B = 59.869 \cdot \text{ft}$

$T \equiv \frac{V_{FL}}{C_P \cdot C_X \cdot LWL \cdot B}$ $T = 20.461 \cdot \text{ft}$ $D_{10} \equiv \frac{LWL}{C_{D10}}$ $D_{10} = 41.805 \cdot \text{ft}$ $V_D \equiv 192009 \cdot \text{ft}^3$

Balance Check

	Required/Minimal	**Available**	**Error**
Weight:	$W_T = 8101.112 \cdot \text{lton}$	$W_{FL} = 8127 \cdot \text{lton}$	$ERR = 0.003$
Propulsion power:	$P_{IREQ} = 103093.704 \cdot \text{hp}$	$P_I = 103100 \cdot \text{hp}$	$ERR_V = 0$
Electrical plant:	$KW_{GREQ} = 1931.155 \cdot \text{kW}$	$KW_G = 2500 \cdot \text{kW}$	
Mach. box height:	$H_{MBMIN} = 22 \cdot \text{ft}$	$H_{MB} = 31.145 \cdot \text{ft}$	
Depth:	$D_{10MIN} = 33.033 \cdot \text{ft}$	$D_{10} = 41.805 \cdot \text{ft}$	
Deckhouse limits:	$V_{DMAX} = 367357.909 \cdot \text{ft}^3$	$V_D = 192009 \cdot \text{ft}^3$	
Deckhouse area:	$A_{DR} = 10207.142 \cdot \text{ft}^2$	$A_{DA} = 18012.101 \cdot \text{ft}^2$	
Deckhouse volume:	$V_{DR} = 108808.139 \cdot \text{ft}^3$	$V_D = 192009 \cdot \text{ft}^3$	
Total ship area:	$A_{TR} = 66772.943 \cdot \text{ft}^2$	$A_{TA} = 66738.347 \cdot \text{ft}^2$	$ERR_A = -0.001$
Total ship volume:	$V_{TR} = 711799.576 \cdot \text{ft}^3$	$V_{TA} = 711430.781 \cdot \text{ft}^3$	
Sustained speed:	**29.96 knt**	$V_S = 30 \cdot \text{knt}$	$V_{MAX} = 31.581 \cdot \text{knt}$
Initial stability:	**0.090-0.135**	$C_{GMB} = 0.105$	

Residuary Drag Coefficients Module

This module calculates Gertler residuary drag coefficients for the use of DDGx design Model. It reads data tables from several data files which must exist on the same directory.

First the data files are loaded into the module. The parameter k represents the coordinates of C_P, so the right data files will be read:

```
k := | i ← 0
     | while $C_V > (i + 1)\cdot 0.001$
     |   i ← i + 1
     | i
```

```
RESID1 := | READPRN(Cv1) if k=1
          | READPRN(Cv2) if k=2
          | READPRN(Cv3) if k=3
          | READPRN(Cv4) if k=4
          | READPRN(Cv5) if k=5
          | READPRN(Cv6) otherwise

RESID2 := | READPRN(Cv2) if k=1
          | READPRN(Cv3) if k=2
          | READPRN(Cv4) if k=3
          | READPRN(Cv5) if k=4
          | READPRN(Cv6) otherwise
```

Next, the coordinates for the coefficients are found:

$$RR_i := \frac{R_i}{\left(\frac{knt}{\sqrt{ft}}\right)}$$

```
m_i := | l ← 0
       | while $RR_i > 0.5 + 0.1\cdot l$
       |   l ← l + 1
       | l

n := | l ← 0
     | while $C_P > 0.48 + 0.02\cdot l$
     |   l ← l + 1
     | l
```

Now, values are interpolated for each B/T ratio (2.25/3.00/3.75) and each straddling C_P:

```
$C1_{R2.25_i}$ := | X1 ← $RESID1_{n,m_i} + \frac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID1_{n+1,m_i} - RESID1_{n,m_i}\right)$
                 | X2 ← $RESID1_{n,m_i+1} + \frac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID1_{n+1,m_i+1} - RESID1_{n,m_i+1}\right)$
                 | $X1 + \frac{RR_i - 0.5 - (m_i - 1)\cdot 0.1}{0.1}\cdot(X2 - X1)$
```

$$C2_{R2.25_i} := \left| \begin{array}{l} X1 \leftarrow RESID2_{n,m_i} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID2_{n+1,m_i} - RESID2_{n,m_i}\right) \\ X2 \leftarrow RESID2_{n,m_i+1} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID2_{n+1,m_i+1} - RESID2_{n,m_i+1}\right) \\ X1 + \dfrac{RR_i - 0.5 - (m_i - 1)\cdot 0.1}{0.1}\cdot(X2 - X1) \end{array} \right.$$

$$C_{R2.25_i} := C1_{R2.25_i} + \frac{C_V - k\cdot 0.001}{0.001}\cdot\left(C2_{R2.25_i} - C1_{R2.25_i}\right) \qquad C_{R2.25_i} := \frac{C_{R2.25_i}}{1000}$$

$$C1_{R3.00_i} := \left| \begin{array}{l} X1 \leftarrow RESID1_{n+12,m_i} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID1_{n+13,m_i} - RESID1_{n+12,m_i}\right) \\ X2 \leftarrow RESID1_{n+12,m_i+1} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID1_{n+13,m_i+1} - RESID1_{n+12,m_i+1}\right) \\ X1 + \dfrac{RR_i - 0.5 - (m_i - 1)\cdot 0.1}{0.1}\cdot(X2 - X1) \end{array} \right.$$

$$C2_{R3.00_i} := \left| \begin{array}{l} X1 \leftarrow RESID2_{n+12,m_i} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID2_{n+13,m_i} - RESID2_{n+12,m_i}\right) \\ X2 \leftarrow RESID2_{n+12,m_i+1} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID2_{n+13,m_i+1} - RESID2_{n+12,m_i+1}\right) \\ X1 + \dfrac{RR_i - 0.5 - (m_i - 1)\cdot 0.1}{0.1}\cdot(X2 - X1) \end{array} \right.$$

$$C_{R3.00_i} := C1_{R3.00_i} + \frac{C_V - k\cdot 0.001}{0.001}\cdot\left(C2_{R3.00_i} - C1_{R3.00_i}\right) \qquad C_{R3.00_i} := \frac{C_{R3.00_i}}{1000}$$

$$C1_{R3.75_i} := \left| \begin{array}{l} X1 \leftarrow RESID1_{n+24,m_i} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID1_{n+25,m_i} - RESID1_{n+24,m_i}\right) \\ X2 \leftarrow RESID1_{n+24,m_i+1} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID1_{n+25,m_i+1} - RESID1_{n+24,m_i+1}\right) \\ X1 + \dfrac{RR_i - 0.5 - (m_i - 1)\cdot 0.1}{0.1}\cdot(X2 - X1) \end{array} \right.$$

$$C2_{R3.75_i} := \left| \begin{array}{l} X1 \leftarrow RESID2_{n+24,m_i} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID2_{n+25,m_i} - RESID2_{n+24,m_i}\right) \\ X2 \leftarrow RESID2_{n+24,m_i+1} + \dfrac{C_P - 0.48 - (n-1)\cdot 0.02}{0.02}\cdot\left(RESID2_{n+25,m_i+1} - RESID2_{n+24,m_i+1}\right) \\ X1 + \dfrac{RR_i - 0.5 - \left(m_i - 1\right)\cdot 0.1}{0.1}\cdot(X2 - X1) \end{array} \right.$$

$$C_{R3.75_i} := C1_{R3.75_i} + \frac{C_V - k\cdot 0.001}{0.001}\cdot\left(C2_{R3.75_i} - C1_{R3.75_i}\right) \qquad C_{R3.75_i} := \frac{C_{R3.75_i}}{1000}$$

Now the values for the residuary drag coefficients are returned to the model, and the resistance calculation continues.

Appendix E

Extrapolated Residuary Drag Coefficients

The DDGx model employs Gertler's reanalysis of Taylor's residuary drag coefficients [20] for its resistance and powering calculations. Original data in [20] is provided in tables; numeric data sorted in arrays is taken from [14]. Section 2.5.2 describes the residuary drag computation in details. Since Gertler has conducted his experiments for a limited parameter range, as shown in Table 2.1, several extrapolation rules were considered, all of them discussed also in Section 2.5.2. The final adopted extrapolation rule relies on ASSET algorithm, with a negligible departure for the highest volumetric coefficient of 0.006.

Data is organized as follows. Tables D.1 through D.6 show the coefficients at volumetric coefficients of 0.001 through 0.006, respectively. Each picture includes three blocks of data, corresponding to beam to draft ratios of 2.25, 3.00 and 3.75, respectively. Every block stores coefficients as a function of the prismatic coefficient and speed to length ratio. Columns represent constant speed to length ratio, while rows represent fixed prismatic coefficient. The speed to length ratio varies between 0.5 to 2.0 knot/ft$^{0.5}$, in jumps of 0.1 knot/ft$^{0.5}$. The prismatic coefficient varies between 0.48 and 0.70 in steps of 0.02. Note that none of these values is shown in the tables - just remember that the upper left cell of each block corresponds to a speed to length ratio of 0.5 knot/ft$^{0.5}$ and prismatic coefficient of 0.48. All data is multiplied by 1000 (i.e., the computer divides them by 1000). Original experimental results are shadowed. These tables are stored as library files from which the resistance module reads and interpolates. In an event of deviation from the specified speed to length or prismatic coefficient domains, end extrapolation is performed.

0.21	0.21	0.21	0.21	0.21	0.29	0.60	1.15	1.77	2.19	2.38	2.40	2.38	2.30	2.20	2.10
0.20	0.20	0.20	0.20	0.20	0.25	0.49	0.97	1.55	1.94	2.13	2.20	2.18	2.11	2.03	1.92
0.20	0.20	0.21	0.21	0.22	0.26	0.46	0.88	1.38	1.73	1.91	2.00	2.00	1.98	1.91	1.86
0.22	0.22	0.22	0.22	0.22	0.26	0.37	0.75	1.20	1.57	1.80	1.88	1.89	1.85	1.79	1.70
0.22	0.22	0.22	0.22	0.24	0.30	0.40	0.71	1.12	1.47	1.68	1.75	1.77	1.73	1.68	1.60
0.22	0.22	0.22	0.22	0.28	0.38	0.41	0.65	1.06	1.43	1.63	1.71	1.72	1.70	1.66	1.59
0.23	0.23	0.23	0.25	0.32	0.45	0.48	0.65	1.03	1.37	1.57	1.66	1.69	1.68	1.62	1.56
0.23	0.23	0.23	0.24	0.38	0.54	0.59	0.73	1.07	1.37	1.54	1.64	1.67	1.66	1.60	1.53
0.23	0.23	0.23	0.27	0.43	0.70	0.74	0.80	1.10	1.38	1.54	1.63	1.68	1.67	1.60	1.52
0.24	0.24	0.24	0.30	0.50	0.82	0.88	0.88	1.15	1.40	1.56	1.63	1.66	1.63	1.60	1.52
0.23	0.23	0.23	0.33	0.53	0.94	1.05	0.98	1.19	1.41	1.55	1.63	1.67	1.65	1.60	1.55
0.24	0.24	0.24	0.40	0.63	1.11	1.21	1.13	1.28	1.43	1.57	1.63	1.65	1.65	1.60	1.56

0.25	0.25	0.25	0.27	0.28	0.34	0.66	1.20	1.73	2.18	2.40	2.45	2.44	2.39	2.30	2.17
0.25	0.25	0.25	0.32	0.38	0.42	0.69	1.10	1.60	2.06	2.28	2.35	2.32	2.25	2.17	2.11
0.26	0.26	0.27	0.39	0.46	0.48	0.66	1.02	1.46	1.94	2.13	2.17	2.17	2.13	2.08	2.00
0.27	0.27	0.27	0.40	0.51	0.51	0.69	0.97	1.38	1.83	2.03	2.10	2.09	2.04	1.96	1.93
0.28	0.28	0.28	0.42	0.57	0.62	0.67	0.88	1.32	1.71	1.90	1.96	1.97	1.95	1.90	1.86
0.30	0.30	0.30	0.43	0.58	0.68	0.71	0.89	1.29	1.63	1.81	1.90	1.92	1.90	1.86	1.80
0.30	0.30	0.31	0.45	0.59	0.70	0.76	0.90	1.23	1.58	1.77	1.83	1.87	1.85	1.80	1.73
0.30	0.31	0.32	0.42	0.60	0.77	0.81	0.90	1.20	1.54	1.72	1.79	1.80	1.80	1.74	1.70
0.33	0.33	0.33	0.40	0.62	0.85	0.90	0.95	1.19	1.50	1.70	1.75	1.76	1.74	1.70	1.65
0.35	0.35	0.36	0.40	0.63	0.90	1.00	1.02	1.25	1.53	1.67	1.75	1.77	1.76	1.70	1.63
0.34	0.34	0.35	0.40	0.64	1.05	1.17	1.12	1.25	1.53	1.69	1.74	1.76	1.75	1.70	1.62
0.33	0.35	0.36	0.40	0.65	1.20	1.37	1.22	1.25	1.53	1.68	1.74	1.76	1.75	1.70	1.61

0.30	0.30	0.30	0.41	0.48	0.47	0.64	1.20	1.87	2.35	2.68	2.80	2.80	2.76	2.70	2.60
0.33	0.33	0.33	0.42	0.48	0.47	0.65	1.09	1.63	2.13	2.45	2.59	2.60	2.58	2.50	2.40
0.35	0.35	0.35	0.43	0.46	0.44	0.59	0.93	1.48	2.01	2.26	2.37	2.39	2.37	2.31	2.24
0.37	0.37	0.37	0.41	0.47	0.45	0.58	0.87	1.34	1.82	2.06	2.17	2.20	2.20	2.13	2.06
0.40	0.40	0.40	0.43	0.48	0.49	0.59	0.82	1.24	1.65	1.90	2.00	2.06	2.05	2.00	1.91
0.40	0.40	0.40	0.42	0.48	0.54	0.63	0.83	1.23	1.60	1.80	1.93	1.97	1.97	1.91	1.78
0.42	0.42	0.42	0.45	0.52	0.61	0.73	0.81	1.18	1.53	1.74	1.87	1.93	1.90	1.85	1.73
0.42	0.42	0.42	0.44	0.54	0.69	0.81	0.90	1.16	1.53	1.77	1.89	1.90	1.89	1.80	1.70
0.45	0.45	0.45	0.50	0.58	0.80	0.93	1.00	1.18	1.52	1.73	1.83	1.88	1.85	1.78	1.69
0.47	0.47	0.47	0.56	0.67	0.90	1.05	1.09	1.23	1.53	1.74	1.85	1.88	1.84	1.77	1.67
0.48	0.48	0.48	0.62	0.70	1.00	1.20	1.20	1.30	1.52	1.71	1.85	1.87	1.82	1.74	1.69
0.49	0.49	0.49	0.68	0.73	1.10	1.34	1.33	1.37	1.52	1.71	1.84	1.87	1.80	1.72	1.70

Table D.1. *Extrapolated residuary drag coefficients (×1000), C_F=0.001.*

0.25	0.25	0.26	0.35	0.41	0.47	1.03	2.23	3.73	4.84	5.40	5.53	5.51	5.39	5.07	4.53
0.25	0.27	0.28	0.31	0.38	0.43	0.89	1.90	3.29	4.43	5.05	5.25	5.26	5.12	4.83	4.37
0.26	0.26	0.27	0.30	0.37	0.42	0.82	1.74	3.00	4.16	4.80	5.02	5.05	4.90	4.60	4.22
0.28	0.28	0.28	0.30	0.40	0.47	0.73	1.54	2.71	3.84	4.55	4.80	4.80	4.68	4.40	4.10
0.28	0.28	0.28	0.30	0.45	0.60	0.78	1.45	2.55	3.65	4.33	4.58	4.60	4.48	4.24	3.92
0.28	0.28	0.28	0.32	0.51	0.77	0.88	1.40	2.45	3.48	4.12	4.38	4.39	4.30	4.10	3.85
0.28	0.28	0.28	0.37	0.59	1.00	1.03	1.41	2.34	3.33	3.90	4.17	4.20	4.13	3.95	3.80
0.28	0.28	0.28	0.40	0.68	1.23	1.20	1.54	2.38	3.28	3.81	4.04	4.10	4.02	3.86	3.62
0.30	0.30	0.30	0.43	0.77	1.48	1.49	1.70	2.41	3.19	3.73	3.97	3.99	3.95	3.81	3.60
0.28	0.29	0.32	0.51	0.89	1.72	1.83	1.89	2.46	3.23	3.69	3.90	3.95	3.90	3.79	3.64
0.28	0.28	0.32	0.60	0.98	1.97	2.10	2.07	2.52	3.24	3.68	3.87	3.92	3.89	3.78	3.61
0.30	0.30	0.37	0.70	1.11	2.22	2.44	2.28	2.64	3.30	3.68	3.87	3.93	3.90	3.77	3.60

0.29	0.30	0.32	0.50	0.57	0.60	1.13	2.19	3.60	4.90	5.56	5.80	5.76	5.57	5.25	5.05
0.30	0.30	0.31	0.51	0.62	0.60	1.04	1.93	3.29	4.57	5.31	5.50	5.50	5.40	5.20	4.80
0.33	0.33	0.33	0.52	0.66	0.66	0.97	1.75	3.06	4.27	5.00	5.30	5.30	5.25	5.10	4.90
0.32	0.32	0.33	0.52	0.68	0.71	0.96	1.61	2.80	4.04	4.72	5.00	5.00	5.00	4.80	4.70
0.34	0.34	0.34	0.54	0.73	0.82	1.00	1.60	2.68	3.80	4.42	4.71	4.80	4.77	4.64	4.40
0.34	0.34	0.37	0.55	0.78	0.98	1.15	1.65	2.57	3.62	4.26	4.40	4.50	4.50	4.40	4.20
0.37	0.38	0.40	0.57	0.83	1.17	1.34	1.70	2.55	3.52	4.17	4.45	4.50	4.55	4.50	4.30
0.38	0.38	0.40	0.58	0.88	1.35	1.60	1.85	2.55	3.46	4.08	4.34	4.42	4.42	4.35	4.20
0.38	0.38	0.41	0.59	0.94	1.60	1.85	2.02	2.54	3.40	4.05	4.41	4.55	4.50	4.30	4.00
0.40	0.40	0.48	0.63	1.02	1.82	2.07	2.21	2.70	3.44	4.03	4.27	4.35	4.25	4.10	3.80
0.40	0.40	0.48	0.70	1.11	2.07	2.40	2.45	2.80	3.48	4.04	4.28	4.35	4.30	4.10	3.80
0.41	0.41	0.54	0.80	1.24	2.30	2.70	2.71	2.98	3.60	4.05	4.26	4.32	4.30	4.10	3.80

0.35	0.35	0.40	0.53	0.67	0.65	1.13	2.30	3.72	5.00	5.95	6.40	6.50	6.40	6.10	5.50
0.38	0.38	0.42	0.54	0.65	0.63	1.00	1.95	3.40	4.67	5.50	5.80	5.90	5.80	5.50	5.10
0.41	0.41	0.43	0.53	0.60	0.63	0.89	1.78	3.10	4.33	5.10	5.35	5.40	5.30	5.10	4.90
0.43	0.43	0.45	0.53	0.64	0.70	0.91	1.70	2.90	4.07	4.79	5.00	5.10	5.10	4.90	4.70
0.45	0.45	0.45	0.53	0.68	0.81	0.97	1.62	2.73	3.87	4.60	4.95	5.10	5.00	4.90	4.60
0.45	0.45	0.47	0.53	0.72	0.91	1.15	1.59	2.62	3.75	4.48	4.90	5.00	5.00	4.90	4.50
0.48	0.49	0.50	0.56	0.78	1.10	1.35	1.78	2.64	3.70	4.40	4.74	4.80	4.75	4.60	4.25
0.48	0.48	0.50	0.56	0.84	1.30	1.60	1.92	2.63	3.65	4.34	4.65	4.70	4.58	4.40	4.10
0.52	0.52	0.53	0.63	0.93	1.53	1.85	2.13	2.77	3.63	4.29	4.61	4.70	4.55	4.35	4.00
0.53	0.54	0.56	0.70	1.01	1.78	2.10	2.32	2.85	3.61	4.24	4.55	4.60	4.50	4.25	3.90
0.55	0.55	0.58	0.79	1.16	2.05	2.40	2.57	3.00	3.67	4.23	4.53	4.55	4.40	4.10	3.80
0.58	0.58	0.66	0.90	1.30	2.30	2.73	2.83	3.20	3.70	4.22	4.50	4.45	4.25	4.00	3.70

Table D.2. *Extrapolated residuary drag coefficients (×1000), C_F = 0.002.*

0.31	0.31	0.33	0.50	0.57	0.64	1.50	3.40	5.65	7.33	8.18	8.38	8.35	8.16	7.68	6.86
0.31	0.31	0.34	0.43	0.52	0.60	1.20	1.80	5.10	6.87	7.83	8.14	8.15	7.94	7.49	6.77
0.30	0.30	0.32	0.38	0.48	0.60	1.12	2.60	4.60	6.38	7.36	7.70	7.74	7.51	7.05	6.47
0.33	0.33	0.34	0.39	0.52	0.78	1.07	2.23	4.17	5.91	7.00	7.39	7.39	7.20	6.77	6.31
0.33	0.33	0.35	0.40	0.58	0.86	1.13	2.10	3.90	5.58	6.62	7.00	7.04	6.85	6.48	6.00
0.34	0.34	0.34	0.42	0.67	1.14	1.30	2.05	3.83	5.44	6.44	6.85	6.86	6.72	6.41	6.02
0.34	0.34	0.35	0.46	0.79	1.45	1.55	2.10	3.67	5.22	6.12	6.54	6.59	6.48	6.20	5.96
0.34	0.34	0.37	0.50	0.89	1.76	1.84	2.26	3.65	5.03	5.84	6.20	6.29	6.17	5.92	5.55
0.34	0.34	0.38	0.57	1.00	2.09	2.23	2.50	3.68	4.87	5.70	6.06	6.09	6.03	5.82	5.50
0.33	0.33	0.40	0.65	1.15	2.43	2.69	2.82	3.80	4.99	5.70	6.02	6.10	6.02	5.85	5.62
0.34	0.34	0.41	0.74	1.29	2.75	3.18	3.18	3.90	5.01	5.70	5.99	6.07	6.02	5.85	5.59
0.35	0.36	0.47	0.85	1.45	3.12	3.76	3.63	4.10	5.12	5.72	6.01	6.10	6.06	5.85	5.59

0.36	0.36	0.37	0.66	0.78	0.86	1.60	3.40	6.10	8.30	9.42	9.83	9.76	9.44	8.90	8.56
0.35	0.35	0.38	0.62	0.76	0.82	1.40	2.90	5.10	7.08	8.23	8.53	8.53	8.37	8.06	7.44
0.38	0.38	0.39	0.60	0.77	0.81	1.25	2.55	4.75	6.63	7.76	8.23	8.23	8.15	7.92	7.61
0.38	0.38	0.40	0.60	0.78	0.84	1.23	2.45	4.45	6.42	7.50	7.95	7.95	7.95	7.63	7.47
0.40	0.40	0.42	0.60	0.82	1.00	1.30	2.34	4.20	5.96	6.93	7.38	7.52	7.48	7.27	6.90
0.40	0.40	0.43	0.62	0.89	1.25	1.50	2.36	3.97	5.59	6.58	6.80	6.95	6.95	6.80	6.49
0.42	0.42	0.44	0.64	0.99	1.52	1.80	2.50	3.83	5.29	6.26	6.68	6.76	6.83	6.76	6.46
0.44	0.44	0.47	0.68	1.08	1.83	2.20	2.70	3.90	5.29	6.24	6.64	6.76	6.76	6.65	6.42
0.43	0.43	0.49	0.72	1.20	2.17	2.63	3.01	3.98	5.33	6.35	6.91	7.13	7.05	6.74	6.27
0.45	0.46	0.55	0.79	1.35	2.50	3.05	3.34	4.20	5.35	6.27	6.64	6.77	6.61	6.38	5.91
0.46	0.46	0.58	0.94	1.50	2.85	3.54	3.75	4.45	5.53	6.42	6.80	6.91	6.83	6.52	6.04
0.48	0.49	0.67	1.07	1.65	3.24	4.04	4.18	4.80	5.80	6.52	6.86	6.96	6.93	6.60	6.12

0.42	0.42	0.49	0.70	0.88	0.95	1.68	3.43	6.00	8.06	9.60	10.32	10.48	10.32	9.84	8.87
0.46	0.46	0.50	0.67	0.81	0.88	1.42	2.91	5.40	7.42	8.74	9.21	9.37	9.21	8.74	8.10
0.46	0.46	0.51	0.66	0.77	0.85	1.30	2.65	4.95	6.91	8.14	8.54	8.62	8.46	8.14	7.82
0.50	0.50	0.54	0.67	0.77	0.96	1.28	2.44	4.45	6.25	7.35	7.67	7.83	7.83	7.52	7.21
0.50	0.50	0.53	0.64	0.81	1.08	1.38	2.33	4.10	5.81	6.91	7.43	7.66	7.51	7.36	6.91
0.52	0.52	0.54	0.63	0.88	1.24	1.60	2.45	4.05	5.80	6.93	7.57	7.73	7.73	7.57	6.96
0.53	0.55	0.57	0.65	0.97	1.47	1.87	2.53	4.10	5.75	6.83	7.36	7.45	7.38	7.14	6.60
0.53	0.53	0.58	0.66	1.08	1.75	2.25	2.90	4.15	5.76	6.85	7.34	7.42	7.23	6.94	6.47
0.58	0.58	0.62	0.73	1.24	2.10	2.65	3.17	4.30	5.64	6.66	7.16	7.30	7.06	6.75	6.21
0.60	0.60	0.67	0.84	1.31	2.48	3.10	3.53	4.50	5.70	6.69	7.18	7.26	7.11	6.71	6.16
0.62	0.62	0.68	0.95	1.48	2.86	3.60	3.96	4.73	5.79	6.67	7.14	7.17	6.94	6.46	5.99
0.63	0.63	0.77	1.10	1.64	3.27	4.13	4.38	5.07	5.86	6.69	7.13	7.05	6.73	6.34	5.86

Table D.3. *Extrapolated residuary drag coefficients (×1000), C_L=0.003.*

0.35	0.35	0.40	0.60	0.72	0.85	2.00	4.70	7.70	9.99	11.15	11.42	11.37	11.13	10.47	9.35
0.35	0.36	0.39	0.50	0.60	0.74	1.54	3.65	7.00	9.43	10.74	11.17	11.19	10.89	10.28	9.30
0.35	0.35	0.38	0.46	0.57	0.73	1.37	3.30	6.30	8.74	10.08	10.54	10.61	10.29	9.66	8.86
0.38	0.38	0.40	0.47	0.61	0.83	1.37	2.85	5.70	8.08	9.57	10.10	10.10	9.84	9.25	8.62
0.38	0.38	0.40	0.48	0.70	1.10	1.49	2.72	4.32	6.18	7.34	7.76	7.79	7.59	7.18	6.64
0.38	0.38	0.40	0.51	0.83	1.45	1.70	2.65	5.20	7.39	8.74	9.30	9.32	9.13	8.70	8.17
0.38	0.38	0.42	0.55	0.97	1.80	2.00	2.70	4.00	5.69	6.67	7.13	7.18	7.06	6.75	6.50
0.39	0.39	0.43	0.62	1.10	2.18	2.43	2.97	5.10	7.03	8.16	8.66	8.79	8.61	8.27	7.76
0.40	0.40	0.46	0.69	1.20	2.62	2.95	3.30	5.00	6.62	7.74	8.24	8.28	8.20	7.90	7.47
0.40	0.40	0.48	0.77	1.40	3.07	3.56	3.73	5.15	6.76	7.72	8.16	8.27	8.16	7.93	7.62
0.40	0.40	0.50	0.86	1.55	3.50	4.23	4.23	5.30	6.81	7.74	8.14	8.24	8.18	7.95	7.59
0.40	0.42	0.57	0.98	1.71	4.00	5.03	4.89	5.60	7.00	7.81	8.21	8.34	8.27	8.00	7.64

0.41	0.41	0.45	0.76	0.95	1.07	2.07	4.75	9.20	12.52	14.21	14.82	14.72	14.23	13.42	12.91
0.42	0.42	0.45	0.72	0.86	0.98	1.80	3.95	7.00	9.72	11.30	11.70	11.70	11.49	11.06	10.21
0.44	0.44	0.47	0.68	0.83	0.96	1.58	3.50	6.50	9.07	10.62	11.26	11.26	11.15	10.83	10.41
0.43	0.43	0.47	0.65	0.82	0.97	1.52	3.13	6.20	8.95	10.45	11.07	11.07	11.07	10.63	10.41
0.44	0.45	0.48	0.64	0.89	1.15	1.58	3.00	5.54	7.86	9.14	9.74	9.92	9.86	9.59	9.10
0.47	0.47	0.50	0.66	0.95	1.48	1.83	2.94	5.20	7.32	8.62	8.90	9.11	9.11	8.90	8.50
0.47	0.47	0.52	0.70	1.05	1.83	2.36	3.12	5.03	6.94	8.23	8.78	8.88	8.98	8.88	8.48
0.50	0.50	0.54	0.73	1.18	2.20	2.77	3.44	5.32	7.22	8.51	9.05	9.22	9.22	9.08	8.76
0.50	0.51	0.58	0.80	1.37	2.67	3.30	3.80	5.40	7.23	8.61	9.38	9.67	9.57	9.14	8.50
0.53	0.53	0.62	0.89	1.50	3.10	3.90	4.27	5.70	7.26	8.51	9.01	9.18	8.97	8.66	8.02
0.53	0.53	0.68	1.04	1.73	3.55	4.62	4.90	6.17	7.67	8.90	9.43	9.59	9.48	9.03	8.37
0.55	0.56	0.78	1.22	1.90	4.00	5.30	5.60	6.60	7.97	8.97	9.43	9.57	9.52	9.08	8.42

0.48	0.50	0.56	0.84	1.09	1.28	2.20	4.63	8.20	11.02	13.12	14.11	14.33	14.11	13.45	12.12
0.52	0.52	0.60	0.82	0.99	1.16	1.90	4.10	7.10	9.75	11.49	12.11	12.32	12.11	11.49	10.65
0.55	0.55	0.59	0.78	0.92	1.10	1.72	3.61	6.50	9.08	10.69	11.22	11.32	11.11	10.69	10.27
0.55	0.55	0.61	0.80	0.92	1.18	1.68	3.30	5.95	8.35	9.83	10.26	10.46	10.46	10.05	9.64
0.57	0.57	0.62	0.73	0.96	1.33	1.77	3.10	5.50	7.80	9.27	9.97	10.27	10.07	9.87	9.27
0.58	0.58	0.62	0.73	1.01	1.51	2.05	3.13	5.30	7.59	9.06	9.91	10.11	10.11	9.91	9.10
0.62	0.60	0.65	0.76	1.11	1.76	2.40	3.30	5.20	7.29	8.67	9.34	9.45	9.36	9.06	8.37
0.62	0.62	0.68	0.78	1.25	2.10	2.80	3.60	5.30	7.36	8.75	9.37	9.47	9.23	8.87	8.26
0.65	0.65	0.73	0.85	1.40	2.50	3.30	3.98	5.55	7.27	8.60	9.24	9.42	9.12	8.72	8.01
0.67	0.67	0.76	0.96	1.58	2.95	3.83	4.44	5.80	7.35	8.63	9.26	9.36	9.16	8.65	7.94
0.67	0.67	0.80	1.09	1.75	3.45	4.43	5.00	6.50	7.95	9.16	9.82	9.86	9.53	8.88	8.23
0.72	0.71	0.85	1.23	1.94	3.90	5.15	5.63	6.54	7.56	8.62	9.20	9.09	8.69	8.18	7.56

Table D.4. *Extrapolated residuary drag coefficients (×1000), C_{t}=0.004.*

0.40	0.41	0.48	0.70	0.88	1.10	2.60	6.00	9.70	12.59	14.04	14.38	14.33	14.02	13.18	11.78
0.42	0.42	0.46	0.60	0.72	0.90	1.90	4.45	9.00	12.12	13.81	14.36	14.39	14.01	13.21	11.95
0.42	0.42	0.43	0.52	0.67	0.83	1.60	4.15	8.10	11.23	12.96	13.55	13.64	13.23	12.42	11.39
0.42	0.42	0.45	0.52	0.70	0.94	1.57	3.80	7.20	10.20	12.09	12.75	12.75	12.43	11.69	10.89
0.43	0.43	0.46	0.55	0.71	1.26	1.73	3.40	6.80	9.73	11.55	12.21	12.27	11.95	11.31	10.45
0.44	0.44	0.48	0.58	0.95	1.70	2.00	3.45	6.50	9.23	10.93	11.62	11.65	11.41	10.88	10.21
0.45	0.45	0.50	0.63	1.10	2.11	2.45	3.33	6.35	9.04	10.58	11.32	11.40	11.21	10.72	10.31
0.45	0.45	0.50	0.70	1.26	2.63	3.05	3.64	6.15	8.48	9.85	10.44	10.59	10.39	9.97	9.35
0.47	0.47	0.53	0.78	1.40	3.10	3.70	4.10	5.30	7.02	8.20	8.73	8.77	8.69	8.38	7.92
0.46	0.48	0.57	0.86	1.58	3.70	4.46	4.64	6.40	8.40	9.60	10.15	10.28	10.15	9.86	9.47
0.45	0.46	0.58	0.95	1.75	4.20	5.33	5.28	6.80	8.74	9.93	10.44	10.58	10.50	10.20	9.74
0.47	0.49	0.68	1.08	1.93	4.90	6.30	6.00	7.05	8.81	9.83	10.33	10.49	10.41	10.07	9.61

0.47	0.47	0.53	0.87	1.10	1.29	2.73	6.10	12.00	16.33	18.53	19.33	19.20	18.57	17.50	16.83
0.48	0.48	0.52	0.81	0.97	1.17	2.25	4.90	9.00	12.50	14.53	15.05	15.05	14.77	14.22	13.13
0.50	0.50	0.54	0.75	0.90	1.08	1.90	4.50	8.20	11.44	13.40	14.20	14.20	14.07	13.67	13.13
0.52	0.52	0.54	0.72	0.87	1.07	1.80	3.80	7.50	10.82	12.64	13.39	13.39	13.39	12.86	12.59
0.52	0.52	0.56	0.70	0.93	1.25	1.87	3.60	6.70	9.50	11.05	11.77	12.00	11.92	11.60	11.00
0.53	0.53	0.57	0.73	1.01	1.69	2.21	3.45	6.25	8.80	10.36	10.70	10.94	10.94	10.70	10.21
0.55	0.55	0.58	0.76	1.13	2.13	2.74	3.65	6.10	8.42	9.98	10.65	10.76	10.88	10.76	10.29
0.53	0.53	0.62	0.80	1.28	2.58	3.32	4.04	6.75	9.16	10.80	11.49	11.70	11.70	11.51	11.12
0.56	0.57	0.66	0.87	1.46	3.09	3.94	4.55	6.80	9.10	10.84	11.81	12.18	12.05	11.51	10.71
0.58	0.58	0.73	0.96	1.63	3.60	4.74	5.17	7.20	9.17	10.75	11.39	11.60	11.33	10.93	10.13
0.58	0.58	0.77	1.10	1.84	4.17	5.74	5.95	7.90	9.82	11.40	12.08	12.27	12.13	11.57	10.72
0.60	0.62	0.88	1.31	2.07	4.70	6.45	7.05	8.40	10.15	11.42	12.01	12.18	12.12	11.56	10.71

0.55	0.56	0.65	1.00	1.32	1.60	2.86	5.90	10.40	13.98	16.63	17.89	18.17	17.89	17.05	15.38
0.58	0.58	0.65	0.94	1.16	1.35	2.48	5.15	8.80	12.09	14.24	15.01	15.27	15.01	14.24	13.20
0.61	0.60	0.67	0.90	1.07	1.30	2.20	4.50	7.90	11.03	13.00	13.63	13.76	13.51	13.00	12.49
0.62	0.62	0.70	0.87	1.05	1.36	2.10	4.05	7.40	10.39	12.22	12.76	13.01	13.01	12.50	11.99
0.64	0.64	0.70	0.83	1.10	1.53	2.18	3.80	6.90	9.78	11.63	12.51	12.89	12.64	12.38	11.63
0.64	0.64	0.73	0.81	1.15	1.79	2.43	3.80	6.50	9.30	11.11	12.16	12.40	12.40	12.16	11.16
0.68	0.70	0.75	0.87	1.25	2.30	2.85	3.90	6.30	8.83	10.50	11.31	11.45	11.34	10.98	10.14
0.68	0.68	0.77	0.87	1.37	2.55	3.35	4.27	6.20	8.60	10.23	10.96	11.08	10.80	10.37	9.67
0.70	0.72	0.82	0.98	1.57	3.05	3.98	4.80	6.50	8.52	10.07	10.82	11.03	10.68	10.21	9.39
0.73	0.73	0.84	1.09	1.77	3.60	4.65	5.30	6.80	8.61	10.12	10.86	10.98	10.74	10.14	9.31
0.74	0.75	0.90	1.20	2.00	4.10	5.35	5.90	7.20	8.81	10.15	10.87	10.92	10.56	9.84	9.12
0.77	0.77	0.95	1.34	2.20	4.30	5.85	6.40	8.60	9.94	11.34	12.09	11.96	11.42	10.75	9.94

Table D.5. *Extrapolated residuary drag coefficients (×1000), C_T=0.005.*

0.44	0.45	0.53	0.82	1.08	1.31	3.10	7.15	11.55	14.99	16.72	17.13	17.07	16.70	15.70	14.03
0.46	0.47	0.51	0.67	0.86	1.04	2.20	5.14	10.40	14.01	15.96	16.59	16.63	16.19	15.26	13.81
0.47	0.47	0.48	0.59	0.77	0.94	1.81	4.70	9.17	12.72	14.68	15.35	15.45	14.98	14.07	12.90
0.48	0.48	0.50	0.60	0.80	1.03	1.72	4.16	7.89	11.18	13.25	13.97	13.97	13.62	12.81	11.93
0.50	0.49	0.51	0.63	0.90	1.37	1.88	3.70	7.39	10.58	12.56	13.28	13.34	12.99	12.30	11.36
0.50	0.50	0.53	0.65	1.07	1.87	2.20	3.79	7.15	10.15	12.02	12.78	12.81	12.55	11.97	11.23
0.50	0.50	0.55	0.70	1.23	2.43	2.82	3.84	7.31	10.41	12.18	13.04	13.13	12.91	12.35	11.87
0.50	0.50	0.57	0.76	1.40	3.06	3.55	4.24	7.16	9.87	11.46	12.15	12.32	12.09	11.60	10.88
0.52	0.52	0.60	0.83	1.55	3.60	4.30	4.76	6.15	8.15	9.52	10.14	10.18	10.09	9.73	9.20
0.52	0.53	0.65	0.93	1.74	4.15	5.00	5.20	7.18	9.42	10.77	11.38	11.53	11.38	11.06	10.62
0.52	0.53	0.68	1.03	1.93	4.90	6.22	6.16	7.93	10.20	11.59	12.18	12.34	12.25	11.90	11.36
0.52	0.56	0.76	1.18	2.10	4.80	6.17	5.88	6.91	8.63	9.63	10.12	10.28	10.20	9.86	9.41

0.52	0.52	0.61	1.05	1.32	1.57	3.32	7.42	14.60	19.87	22.55	23.53	23.37	22.60	21.30	20.48
0.53	0.54	0.63	0.94	1.10	1.32	2.54	5.53	10.15	14.10	16.39	16.98	16.98	16.66	16.04	14.81
0.55	0.56	0.62	0.86	1.02	1.20	2.11	5.00	9.11	12.71	14.89	15.78	15.78	15.63	15.19	14.59
0.57	0.57	0.63	0.82	0.93	1.20	2.02	4.26	8.41	12.13	14.18	15.02	15.02	15.02	14.42	14.12
0.58	0.58	0.64	0.80	1.00	1.43	2.14	4.12	7.66	10.87	12.64	13.46	13.73	13.64	13.27	12.58
0.58	0.60	0.65	0.83	1.08	1.87	2.45	3.82	6.92	9.74	11.46	11.84	12.11	12.11	11.84	11.30
0.60	0.60	0.68	0.84	1.22	2.40	3.09	4.11	6.87	9.49	11.25	12.00	12.12	12.26	12.12	11.59
0.62	0.62	0.70	0.87	1.37	2.91	3.74	4.56	7.61	10.33	12.18	12.96	13.20	13.20	12.98	12.54
0.62	0.62	0.75	0.94	1.55	3.50	4.46	5.15	7.70	10.31	12.28	13.38	13.80	13.65	13.04	12.13
0.64	0.66	0.80	1.03	1.75	4.10	5.40	5.89	8.20	10.44	12.24	12.97	13.21	12.90	12.45	11.54
0.67	0.67	0.85	1.17	1.93	4.75	6.54	6.78	9.00	11.19	12.99	13.76	13.98	13.82	13.18	12.21
0.67	0.71	0.96	1.38	2.23	4.90	6.72	7.35	8.76	10.58	11.91	12.52	12.70	12.64	12.05	11.17

0.62	0.64	0.75	1.15	1.56	1.93	3.45	7.12	12.54	16.86	20.06	21.58	21.92	21.58	20.57	18.55
0.65	0.67	0.75	1.10	1.35	1.59	2.92	6.07	10.36	14.24	16.77	17.68	17.98	17.68	16.77	15.55
0.68	0.69	0.75	1.01	1.22	1.46	2.47	5.05	8.87	12.39	14.60	15.31	15.45	15.17	14.60	14.03
0.70	0.70	0.78	0.98	1.16	1.51	2.33	4.50	8.22	11.54	13.57	14.17	14.44	14.44	13.88	13.31
0.72	0.72	0.79	0.94	1.19	1.70	2.42	4.22	7.67	10.87	12.92	13.90	14.32	14.04	13.76	12.92
0.72	0.72	0.81	0.94	1.28	2.00	2.72	4.25	7.26	10.39	12.41	13.59	13.85	13.85	13.59	12.47
0.74	0.76	0.83	0.96	1.14	2.42	3.00	4.10	6.63	9.29	11.05	11.90	12.05	11.93	11.55	10.67
0.77	0.77	0.87	1.00	1.56	3.00	3.94	5.02	7.29	10.12	12.04	12.89	13.04	12.71	12.20	11.38
0.78	0.78	0.91	1.10	1.75	3.50	4.57	5.51	7.46	9.78	11.56	12.42	12.66	12.26	11.72	10.78
0.80	0.80	0.97	1.22	1.94	4.30	5.55	6.33	8.12	10.28	12.09	12.97	13.12	12.83	12.11	11.12
0.80	0.82	1.00	1.32	2.14	4.90	6.39	7.05	8.60	10.53	12.13	12.99	13.05	12.62	11.76	10.90
0.82	0.83	1.05	1.47	2.40	5.10	6.94	7.59	10.20	11.79	13.45	14.34	14.19	13.54	12.75	11.79

Table D.6. *Extrapolated residuary drag coefficients (×1000), C_T=0.006.*

Appendix F

DDGx Evolutionary Fortran 90 Optimization model

The complexity and intensity of the design model, written using Mathcad worksheet, as well as the search algorithms, require a more advanced computing tool. Fortran 90 is a modification of the long known Fortran 77 language, with many enhanced capabilities, most of them improve the interface with the user. The software also incorporates professional mathematical and statistical IMSL™ libraries, applicable as intrinsic functions. In this application two numerical solvers are used, for the power and area balances. Fortran 90 compiler is compatible for Fortran 77 codes. A description of the basics is found in [40].

Computer codes for genetic algorithms exist in the market, as reported in [11], for example. A lot of effort has been made to create a comfortable interface in the DDGx code. Its structure is very modular, allowing the addition of new operators or updated search parameters. The code is used for both unconstrained and constrained exploration. Based on the terminology and methods discussed in Chapters 3, 4 and 5, the program asks the user for the required search process, as shown in Figure F.1. In this illustration the exploration involves randomly population initialization of 30 individuals, a maximum of 100 generations, a rank-based selection mechanism with uniform selective pressure, one-point crossover and non-uniform mutation at an update rate of 0.3. No penalty is exerted.

```
Population initialization (y/n/f) ... y
Population size ... 30
Total number of generations ... 100
Selection algorithm (c/r/b) ... r
Selective pressure (u/n) ... u
Crossover splits (0/1/2) ... 1
Mutation type (u/n) ... n
Mutation probability ... 0.3
Penalty function active (y/n) ... n
```

Figure F.1. *DDGx optimization code interface window.*

After the required data is entered, the computer begins with the exploration and optimization procedure. It prints the current generation and chromosome number, and at the completion of any generation, before starting to evaluate the next one, writes the information shown in Figure F.2. The printed data enables control on the progress of the process.

```
0 Duplicated chromosomes.
6 crossovers.
5 mutations.
2 feasible chromosomes.
Best ever so far ...    .1207
Feasible best ever so far ...    .1109
Convergence counter ...    0
```

Figure F.2. *DDGx optimization code intermediate report.*

The program may read separately from two input files, depending on the population initialization method, and writes to three output files. There are other two initialization alternatives. In a forced initialization, a pre-defined population, from the file *DDG_forced.in* is processed. This is of interest when the performance of two algorithms is compared: it eliminates any possible advantage of one algorithm due to a randomly created better start point, say a super-individual. The third option is without initialization. In this case only one chromosome, its genes stored in *par.in* data file, is evaluated, so a complete design report of that ship can be read. This option is needed after an optimal solution has been found by the computer: a complete naval architectural record is given only for the last resolved chromosome, which is not necessarily the optimum.

The program writes to three different output files. A generation-based report, containing average and maximum payload fractions at each generation, is written into *DDG_report.out* file. From this file the fitness versus generation curves are constructed. An example is shown in Figure F.3.

```
Gen   F_av_feas   max_gen_feas    F_av    max_gen   span*100.0   Dupl   Feasible

 1     .1089         .1109       .1073    .1207       6.6065       0       2
 2     .1143         .1204       .1126    .1207       4.6415      14       3
 3     .1195         .1204       .1150    .1207       5.1991      20       2
 4     .1185         .1204       .1171    .1268       4.7688      15       2
 5     .1152         .1204       .1180    .1268       5.5182      13       2
 6     .0000         .0000       .1180    .1268       5.5124       7       0
 7     .0000         .0000       .1206    .1272       4.4213      12       0
 8     .0000         .0000       .1233    .1273       4.3949      12       0
 9     .0000         .0000       .1253    .1273       3.3832      14       0
10     .0000         .0000       .1265    .1273       1.1208      18       0
11     .0000         .0000       .1265    .1275       1.2326      19       0
12     .0000         .0000       .1258    .1275       1.9734      18       0
13     .0000         .0000       .1257    .1275       3.6485      22       0
14     .0000         .0000       .1265    .1275       1.4994      22       0
15     .0000         .0000       .1268    .1275       1.3125      24       0
16     .0000         .0000       .1269    .1275       1.2912      21       0
```

```
 17    .0000        .0000       .1266    .1277      1.4718    14       0
 18    .0000        .0000       .1267    .1277      1.4130    15       0
 19    .0000        .0000       .1266    .1281      1.2545     9       0
 20    .0000        .0000       .1270    .1281      1.0534    18       0
 21    .0000        .0000       .1271    .1281      1.7647    20       0
 22    .0000        .0000       .1273    .1281      1.2054    23       0
 23    .0000        .0000       .1272    .1281      1.0807    21       0
 24    .0000        .0000       .1269    .1281      2.1328    18       0
 25    .0000        .0000       .1273    .1281      1.4574    19       0
 26    .0000        .0000       .1267    .1281      2.0926    16       0
 27    .0000        .0000       .1272    .1281      1.5819    20       0
 28    .0000        .0000       .1273    .1281      1.3076    21       0
 29    .0000        .0000       .1275    .1281      1.2138    20       0
 30    .0000        .0000       .1266    .1281      2.2426    20       0
 31    .0000        .0000       .1270    .1281      2.0711    17       0
 32    .0000        .0000       .1274    .1281      1.2545    21       0
 33    .0000        .0000       .1276    .1281       .6823    22       0
 34    .0000        .0000       .1274    .1281      1.0767    18       0

  Convergence after 34 generations.

  Selection process ... r
  Selective pressure ... u
  Crossover type ...   1
  Mutation type ... n
  Mutation update probability ...   .30
  Penalty function ... n

  Best ever feasible fitness ...    .1204
  Generation# ...  2
  Cp ...                .700
  Cx ...                .890
  Cdl ...              87.31
  Cbt ...              3.40
  CD10 ...             12.00
  Crd ...               .280

  Best ever fitness ...             .1281
  Generation# ... 29
  Cp ...                .700
  Cx ...                .900
  Cdl ...              76.60
  Cbt ...              2.80
  CD10 ...             14.98
  Crd ...               .000
```

Figure F.3. *DDG_report.out sample output file.*

A complete description of all chromosomes evaluated in the exploration is written to *DDG_chrom.out* output file. A sample of this file, for the last generation only, is illustrated in Figures 2.14 and 4.4. In this large file the saturation of the population as the process converges can be seen, as in Figure 4.4. The computer does not provide technical report for every ship explored, because this would create a huge file. Instead, it writes the characteristics of just the very last vessel calculated, into the file *DDG.out*. An example is given in Figure 2.13 for the DDG51, and Figure 5.7 for the thesis optimal ship.

Figure F.4 displays the flowchart of the code.

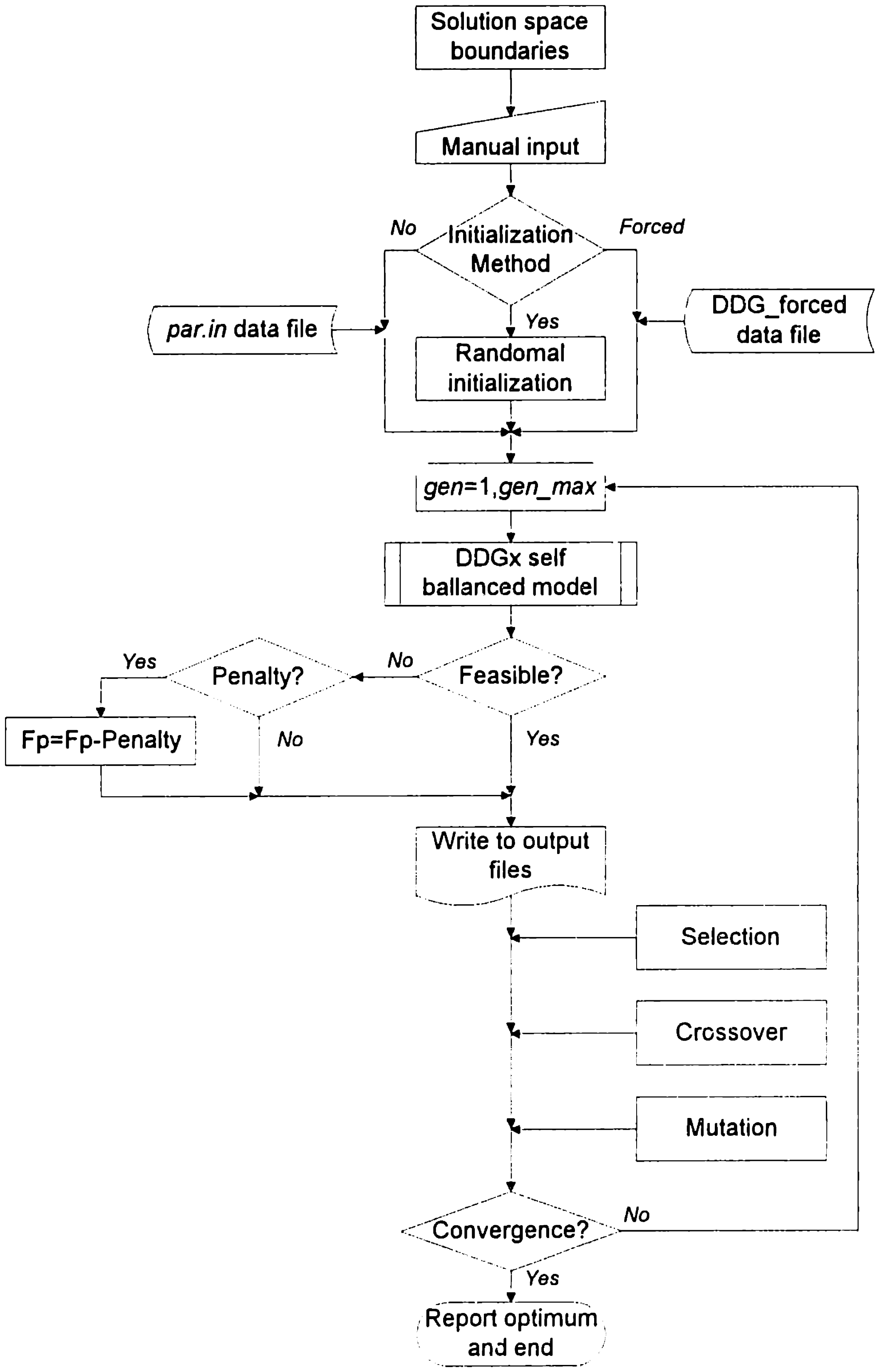


Figure F.4. *DDGx Fortran code flowchart.*

For simplicity, none of the involved subroutines are displayed. The structure of the self-balanced model is illustrated in Figure 2.1. This is especially true in the structure of the genetic operators, which, because of their several alternatives, employ various subroutines. Ship design, particularly in preliminary stages, does not require significantly high accuracy. Therefore, all parameters in the code are declared with their default precision.

The physical unit system used in the evaluation is the U.S. units system. Refer to the Mathcad ship model in Appendix D for more details, as Mathcad tracks the units as it calculates. In all cases the expression in Fortran are converted to the required U.S. units by means of constants. In many cases, even though an expression could have been simplified, it is kept unresolved for clarity.

Internal changes to the code require a Fortran 90 compiler and linker. Note that a trial to run the program on a 66MHz 486 computer at MIT has failed. The CPU could not complete even one generation, and typically stalled after approximately five chromosomes.

A listing of the source file is given on the next page.

```
      program ddg
! Performes genetic algorithm search process for ship design.
! For units refer to Mathcad model; they are in U.S. system.
      use msimsl
          use msflib
          real bal,Vguess(1),VDK(1),LWL,KWgreq,KWg,C_p(30),C_x(30),C_dl(30)
          real C_bt(30),C_D10(30),C_rd(30),F_p(30),max_gen,max_gen_feas
          real max_ever_feas,max_ever,max_gen_prev
          integer info(1),pop_size,gen,gen_max,feas_counter
          character*1 pop_method,select_method,er1,er2,er3,er4,er5,er6,er7,&
            mut_type,pen_act,press_method
          character*2 warning_weight
          external bal
          common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
          common/dimen1/T,B,HDK,D10
          common/dimen/LWL,Vfl
          common/balance/Adr,Ada,KWgreq,KWg,Vs,Cgmb,Fp,warning_weight
          common/genes/C_p,C_x,C_dl,C_bt,C_D10,C_rd
          common/fitness/F_p
          common/raised/Hr
          common/limits/Cpmin,Cpmax,Cxmin,Cxmax,Cdlmin,Cdlmax,Cbtmin,      &
            Cbtmax,CD10min,CD10max,Crdmin,Crdmax
          common/gener/gen,gen_max
! Limits of explored solution space.
          Cpmin=0.5
          Cpmax=0.7
          Cxmin=0.7
          Cxmax=0.9
          Cdlmin=60.0
          Cdlmax=90.0
          Cbtmin=2.8
          Cbtmax=3.7
          CD10min=10.0
          CD10max=15.0
          Crdmin=0.0
          Crdmax=0.8
          open(18,file='c:\thesis\analysis\DDG_report.out',status='replace')       ! File
containing generational report.
          write(18,21) 'Gen','F_av_feas','max_gen_feas','F_av','max_gen',  &
            'span*100.0','Dupl','Feasible'
          write(18,*)
          open(19,file='c:\thesis\analysis\DDG_chrom.out',status='replace') ! File
containing generational report.
! Initialization of population (random or forced genotype).
 101  write(*,14) 'Population initialization (y/n/f) ... '
          read(*,*) pop_method
          if((pop_method.ne.'y').and.(pop_method.ne.'n').and.                     &
            (pop_method.ne.'f')) then
               write(*,3) 'Answer y,n or f.'
               goto 101
          endif
          if(pop_method.eq.'n') then          ! Only one chromosome is evaluated.
               pop_size=1
               gen_max=1
               open(3,file='c:\thesis\analysis\par.in',status='old')
               read(3,*) Cp
               read(3,*) Cx
               read(3,*) Cdl
               read(3,*) Cbt
               read(3,*) CD10
               read(3,*) Crd
               close(3)
               pen_act='n'
          else
               write(*,14) 'Population size ... '
               read(*,*) pop_size
               write(*,14) 'Total number of generations ... '
               read(*,*) gen_max
 115    write(*,14) 'Selection algorithm (c/r/b) ... '
               read(*,*) select_method
               if((select_method.ne.'c').and.(select_method.ne.'r').and.      &
                 (select_method.ne.'b')) then
                      write(*,3) 'Answer c, r, or b.'
                      goto 115
               endif
```

```
116   write(*,14) 'Selective pressure (u/n) ... '
             read(*,*) press_method
             if((press_method.ne.'u').and.(press_method.ne.'n')) then
                   write(*,3) 'Answer u or n.'
                   goto 116
             endif
117   write(*,14) 'Crossover splits (0/1/2) ... '
             read(*,*) nn
             if((nn.ne.0).and.(nn.ne.1).and.(nn.ne.2)) then
                   write(*,3) 'Choose 0, 1 or 2.'
                   goto 117
             endif
118   write(*,14) 'Mutation type (u/n) ... '
             read(*,*) mut_type
             if((mut_type.ne.'u').and.(mut_type.ne.'n')) then
                   write(*,3) 'Choose u for uniform or n for non uniform.'
                   goto 118
             endif
119   write(*,14) 'Mutation probability ... '
             read(*,*) Pup
             if((Pup.le.0.0).or.(Pup.ge.1.0)) then
                   write(*,3) 'Mutation probability out of range.'
                   goto 119
             endif
120   write(*,14) 'Penalty function active (y/n) ... '
             read(*,*) pen_act
             if((pen_act.ne.'y').and.(pen_act.ne.'n')) then
                   write(*,3) 'Answer y or n.'
                   goto 120
             endif
        endif
        write(*,*)
        Pm=1.0-(1.0-Pup)**(1.0/6.0) ! Probability to mutate a single gene.
        max_ever_feas=0.0
        k_feas=0                ! Records the generation of fittest feasible chromosome.
        Cp_ever_feas=0.0
        Cx_ever_feas=0.0
        Cdl_ever_feas=0.0
        Cbt_ever_feas=0.0
        CD10_ever_feas=0.0
        Crd_ever_feas=0.0
        max_ever=0.0
        k_max=0                 ! Records the generation of fittest chromosome.
        Cp_ever=0.0
        Cx_ever=0.0
        Cdl_ever=0.0
        Cbt_ever=0.0
        CD10_ever=0.0
        Crd_ever=0.0
        max_gen_prev=0.0        ! At previous generation.
        if(pop_method.eq.'y') call initial(pop_size)
        if(pop_method.eq.'f') then
             open(3,file='c:\thesis\analysis\DDG_forced.in',status='old')
             do 77 i=1,pop_size
                   read(3,*) C_p(i),C_x(i),C_dl(i),C_bt(i),C_D10(i),C_rd(i)
77           continue
             close(3)
        endif
        do 110 gen=1,gen_max
             write(19,*)
             write(19,13) 'Generation #',gen
             write(19,*)
             write(19,51) 'Cp','Cx','Cdl','Cbt','CD10','Crd','Fp','Area',    &
               'kW','Vs','GM-','GM+','D10','Vd'
             write(19,*)
             F_feas=0.0
             feas_counter=0
             max_gen_feas=0.0
             n_dupl=0
! This 100 loop evaluates the fitness of chromosomes, at generation 'gen'.
             do 100 i=1,pop_size
                   warning_weight='of'              ! Flag for weight convergence status.
                   er1=' '           ! Flag for non sufficient dkhs area.
                   er2=' '           ! Flag for non feasible electric plant.
                   er3=' '           ! Flag for unsatisfactory sustained speed.
```

```
                       er4=' '        ! Flag for poor initial stability.
                       er5=' '        ! Flag for excessive initial stability.
                       er6=' '        ! Flag for non feasible deck height.
                       er7=' '        ! Flag for non feasible dkhs volume.
                       write(*,19) 'Generation #',gen,'-- Chromosome #',i
                       if(pop_method.eq.'n') goto 102
                       Cp=C_p(i)
                       Cx=C_x(i)
                       Cdl=C_dl(i)
                       Cbt=C_bt(i)
                       CD10=C_D10(i)
                       Crd=C_rd(i)
! Accelerate the computation by search for identical chromosome.
                       do 84 m=1,i-1
                               if((Cp.eq.C_p(m)).and.(Cx.eq.C_x(m)).and.                &
                                 (Cdl.eq.C_dl(m)).and.(Cbt.eq.C_bt(m)).and.             &
                                 (CD10.eq.C_D10(m)).and.(Crd.eq.C_rd(m))) then
                                       n_dupl=n_dupl+1
                                       Fp=F_p(m)
                                       write(19,54) Cp,Cx,Cdl,Cbt,CD10,Crd,Fp,
           &
                                         'Duplicated solution - chromosome#',m
                                       write(*,3) 'Duplicated chromosome.'
                                       goto 105
                               endif
 84                    continue
! Iterative solver for deckhouse volume.
 102            Vguess(1)=190000.0     ! Initial guess for deckhouse volume.
                       call zreal(bal,10.0,0.01,0.1,0.1,1,20,Vguess,VDK,info)
                       if((warning_weight.eq.'on').or.(info(1).gt.20)) then
                               Fp=0.0
                               write(19,52) Cp,Cx,Cdl,Cbt,CD10,Crd,Fp
                               goto 105
                       endif
                       write(10,*)
                       VD=VDK(1)
                       write(10,1) 'Deckhouse volume',VD,'[ft^3]'
! Checking balance of ship (other than weight and space).
                       write(10,*)
                       write(10,3) 'Balance/Feasibility Status ...'
                       write(10,*)
                       write(10,4) 'Category','Required/Minimal','Available'
                       write(10,6) 'Deckhouse area [ft^3]',Adr,Ada
                       write(10,8) 'Electric plant [KW]',KWgreq,KWg
                       Vsmin=29.96
                       write(10,5) 'Sustained speed',Vsmin,Vs
                       Cgmbmin=0.09
                       Cgmbmax=0.135
                       write(10,10) 'Initial stability',Cgmbmin,Cgmb
                       write(10,11) Cgmbmax
                       Hmbmin=22.0
                       D10min=max(0.21*B+T,LWL/15,Hmbmin+HDK)
                       if(Crd.ge.0.5) D10min=max(0.21*B+T+Hr,LWL/15+Hr,Hmbmin+HDK)
                       write(10,5) 'Depth [ft]',D10min,D10
                       VDmax=(2*B-4*HDK/tand(80.0))*HDK*0.6*LWL*1.1
                       write(10,7) 'Maximal deckhouse volume [ft^3]',VDmax,VD
                       Fpmin=0.075
                       write(10,10) 'Payload fraction',Fpmin,Fp
                       close(10)
                       if(Ada.lt.Adr) er1='x'
                       if(KWg.lt.KWgreq) er2='x'
                       if(Vs.lt.Vsmin) er3='x'
                       if(Cgmb.lt.Cgmbmin) er4='x'
                       if(Cgmb.gt.Cgmbmax) er5='x'
                       if(D10.lt.D10min) er6='x'
                       if(VD.gt.VDmax) er7='x'
                       if(VD.lt.0.0) er7='-'
                       if((Ada.ge.Adr).and.(KWg.ge.KWgreq).and.(Vs.ge.Vsmin).and.  &
                        (Cgmb.ge.Cgmbmin).and.(Cgmb.le.Cgmbmax).and.               &
                        (D10.ge.D10min).and.(VD.le.VDmax)) then
                               call beepqq(2000,500)
                               write(19,52) Cp,Cx,Cdl,Cbt,CD10,Crd,Fp
                               F_feas=F_feas+Fp
                               feas_counter=feas_counter+1
                               if(Fp.gt.max_gen_feas) max_gen_feas=Fp
```

```
                              if(Fp.gt.max_ever_feas) then
                                     max_ever_feas=Fp
                                     k_feas=gen
                                     Cp_ever_feas=Cp
                                     Cx_ever_feas=Cx
                                     Cdl_ever_feas=Cdl
                                     Cbt_ever_feas=Cbt
                                     CD10_ever_feas=CD10
                                     Crd_ever_feas=Crd
                              endif
                       else
                              if(pen_act.eq.'y') then
                                     ERR_area=0.0
                                     ERR_speed=0.0
                                     ERR_stabil=0.0
                                     ERR_D10=0.0
                                     ERR_VD=0.0
                                     if(er1.eq.'x') ERR_area=(Adr-Ada)/Adr*100
                                     if(er3.eq.'x') ERR_speed=(Vsmin-Vs)/Vsmin*100
                                     if(er4.eq.'x') ERR_stabil=(Cgmbmin-Cgmb)/Cgmbmin*100
                                     if(er5.eq.'x') ERR_stabil=(Cgmb-Cgmbmax)/Cgmbmax*100
                                     if(er6.eq.'x') ERR_D10=(D10min-D10)/D10min*100
                                     if(er7.eq.'x') ERR_VD=(VD-VDmax)/VDmax*100
                                     penalty=0.001*(ERR_area+ERR_speed+ERR_stabil+       &
                                       ERR_D10+ERR_VD)
                                     Fp=Fp-penalty
                              endif
                              write(19,55) Cp,Cx,Cdl,Cbt,CD10,Crd,Fp,er1,er2,er3,er4,&
                                er5,er6,er7
                       endif
                       if(Fp.gt.max_ever) then
                              max_ever=Fp
                              k_max=gen
                              Cp_ever=Cp
                              Cx_ever=Cx
                              Cdl_ever=Cdl
                              Cbt_ever=Cbt
                              CD10_ever=CD10
                              Crd_ever=Crd
                       endif
 105    F_p(i)=Fp
 100    continue
            call beepqq(1000,500)
                if(pop_method.eq.'n') goto 110
                F=sum(F_p)      ! Total fitness of population.
                F_av=F/pop_size        ! Overall average fitness.
                max_gen=maxval(F_p)    ! Maximal payload fraction of generation.
                F_av_feas=0.0
                if(feas_counter.ne.0) F_av_feas=F_feas/feas_counter  ! Average fitness
among feasible genotypes.
                span=sqrt(sum((F_p-F_av)**2)/(pop_size-1))/(F/pop_size)
                write(19,20) gen,F_av_feas,max_gen_feas,F_av,max_gen,span*100.0&
                  ,n_dupl,feas_counter
                if(select_method.eq.'c') call select_class(F,pop_size,gen)
                if((select_method.eq.'r').or.(select_method.eq.'b')) call       &
                  select_rank(pop_size,gen,select_method,press_method)
                call crossover(pop_size,gen,nn,n_cross)
                write(*,*)
                write(*,14) n_dupl,'Duplicated chromosomes.'
                write(*,14) n_cross,'crossovers.'
                call mutation(mut_type,pop_size,Pm,n_mutat)
                write(*,14) n_mutat,'mutations.'
                write(*,14) feas_counter,'feasible chromosomes.'
                write(*,15) 'Best ever so far ... ',max_ever
                write(*,15) 'Feasible best ever so far ... ',max_ever_feas
                if(max_gen_feas.eq.0.0) goto 140
                if(pen_act.eq.'y') then
                       if(nint(1e4*max_gen_feas).eq.nint(1e4*max_gen_prev)) then
                              n_converge=n_converge+1
                       else
                              n_converge=0
                       endif
                       max_gen_prev=max_gen_feas
                endif
 140    if(pen_act.eq.'n') then
```

```
                        if(nint(1e4*max_gen).eq.nint(1e4*max_gen_prev)) then
                                n_converge=n_converge+1
                        else
                                n_converge=0
                        endif
                        max_gen_prev=max_gen
                endif
                write(*,13) 'Convergence counter ... ',n_converge
                write(*,*)
                if(n_converge.eq.15) then
                        write(18,*)
                        write(18,2) 'Convergence after',gen,'generations.'
                        goto 109
                endif
110     continue
109     call beepqq(1000,1000)
            if(pop_method.eq.'n') stop
            write(18,*)
            write(18,13) 'Selection process ...',select_method
            write(18,13) 'Selective pressure ...',press_method
            write(18,13) 'Crossover type ... ',nn
            write(18,13) 'Mutation type ...',mut_type
            write(18,22) 'Mutation update probability ...',Pup
            write(18,13) 'Penalty function ...',pen_act
            write(18,*)
            write(18,15) 'Best ever feasible fitness ...',max_ever_feas
            write(18,13) 'Generation# ...',k_feas
            write(18,16) 'Cp ...',Cp_ever_feas
            write(18,16) 'Cm ...',Cm_ever_feas
            write(18,17) 'Cdl ...',Cdl_ever_feas
            write(18,18) 'Cvt ...',Cvt_ever_feas
            write(18,17) 'CDLL ...',CDLL_ever_feas
            write(18,18) 'Crd ...',Crd_ever_feas
            write(18,*)
            write(18,15) 'Best ever fitness ...',max_ever
            write(18,13) 'Generation# ...',k_max
            write(18,16) 'Cp ...',Cp_ever
            write(18,16) 'Cm ...',Cm_ever
            write(18,17) 'Cdl ...',Cdl_ever
            write(18,18) 'Cvt ...',Cvt_ever
            write(18,17) 'CDLL ...',CDLL_ever
            write(18,18) 'Crd ...',Crd_ever
1       format(1x,a,t45,f4.1,x,a)
2       format(1x,a,x,i2,x,a)
3       format(1x,a)
4       format(1x,a,t40,a,t60,a)
5       format(1x,a,t45,f5.1,t61,f5.1)
6       format(1x,a,t45,f7.1,t61,f7.1)
7       format(1x,a,t45,f4.1,t61,f4.1)
8       format(1x,a,t45,f6.1,t61,f6.1)
9       format(t47,f5.1)
10      format(1x,a,t45,f5.3,t61,f6.4)
11      format(t45,f5.3)
12      format(t45,f5.1)
13      format(1x,a,i3)
14      format(1x,a,\)
15      format(1x,a,t35,f6.4)
16      format(1x,a,t15,f7.3)
17      format(1x,a,t15,f5.1)
18      format(1x,a,t15,f4.1)
19      format(1x,a,i3,x,a,i2)
20      format(i3,t7,f6.4,t18,f6.4,t31,f6.4,t43,f6.4,t5 ,f7.4,t61,i2,t65,a
          i2)
21      format(a,t6,a,t17,a,t31,a,t37,a,t45,a,t51,a,t66,a
22      format(1x,a,x,f4.1
23      format(1x,a,x,a
24      format(1x,i2,x,a
25      format(1x,a,x,f6.4
51      format(1x,a,t7,a,t12,a,t17,a,t22,a,t27,a,t35,a,t41,a,t40,a,t47,a,a
          t54,a,t59,a,t64,a,t69,a
52        format(f4.1,t5,f4.1,t11,f4.1,t17,f5.1,t22,f4.1,t27,f4.1,t32,f6.4)
53        format(f4.1,t5,f4.1,t11,f4.1,t17,f5.1,t22,f4.1,t27,f4.1,t32,f6.4,a
         t42,a,t47,a,t51,a,t55,a,t61,a,t65,a,t71,a
54        format(f4.1,t5,f4.1,t11,f4.1,t17,f5.1,t22,f4.1,t27,f4.1,t32,f6.4,a
         t42,a,i2)
```

```
          stop
          end

          real function bal(VD)
! This function bears the whole model, but actualy is used to calculate the
! area balance error. At each entrance to this function the weight is being balanced,
! and the resulting area error is calculated.
          real LWL,KWg,KW24avg,KG,KWgreq
          character*2 warning_weight
          common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
          common/dimen/LWL,Vfl
          common/dimen1/T,B,HDK,D10
          common/propuls/Npeng,Pbpeng
          common/man/NO,NE,NA
          common/elect/Ng
          common/liquid/WF46,WF52,WF41
          common/weig/Wops
          common/duct/NDIE,NHPIE,NHeIE
          common/balance/Adr,Ada,KWgreq,KWg,Vs,Cgmb,Fp,warning_weight
          Wp=922.9
          Wops=30.0
          NO=26
          NE=315
          NA=36
          Npeng=4
          Pbpeng=25775.0
          Ng=3
          KWg=1500.0
          NDIE=1
          NHPIE=1
          NHeIE=5
! Design requirements.
      HDK=10.66
          Ve=20.0
          E=3907.6
          Ts=45.0
          iter_wei=0               ! Counter of weight iterations.
          Wfl=Wp/0.1
 100  iter_wei=iter_wei+1
      if(iter_wei.gt.15) then
                write(*,3) 'Failed to converge in weight.'
                bal=0.0
                warning_weight='on'
                return
          endif
          open(10,file='c:\thesis\analysis\DDG.out',status='replace')
          write(10,3) 'Main Parameters ...'
          write(10,*)
          write(10,2) 'Cp',Cp
          write(10,5) 'Cx',Cx
          write(10,1) 'Cdl',Cdl,'[lton/ft 3]'
          write(10,5) 'Cbt',Cbt
          write(10,6) 'CD10',CD10
          write(10,5) 'Crd',Crd
! Main dimensions.
      Vfl=Wfl*34.99
          LWL=100*(Wfl/Cdl)**(1./3)
          B=sqrt(Cbt*Vfl/(Cp*Cx*LWL))
          T=Vfl/(Cp*Cx*LWL*B)
          D10=LWL/CD10
          write(10,*)
          write(10,3) 'Main Dimensions on Waterline ...'
          write(10,*)
          write(10,1) 'Underwater volume',Vfl,'[ft 3]'
          write(10,1) 'Length on waterline',LWL,'[ft]'
          write(10,1) 'Beam',B,'[ft]'
          write(10,1) 'Draft',T,'[ft]'
          write(10,1) 'D10',D10,'[ft]'
! Resistance computations.
          call resist(Ve,SHPe,Vs,Vmax)
! Available space calculations.
      call availspace(Cp,Crd,Vht,Cx,Dav)
! Electrical load calculations.
          Pi=Npeng*Pbpeng
          call electric(Vht,VD,Pi,Wops,KW24avg,Vaux,KWgreq)
```

```
! Tankage volumes.
      call tank(E,Ve,Vmax,SHPe,KW24avg,Pi,Vtk)
! Weight.
            Vt=Vht+VD
            call weight(Wfl,Pi,Vt,LWL,CN,Ts,Ng,KWg,VD,Dav,KG,NHPIE,ERRw,Hmb)
            if(ERRw.gt.0.01) then
                  close(10,status='delete')
                  goto 100
            endif
            Fp=Wp/Wfl
            write(10,5) 'Payload fraction',Fp

! Space requirements and balance.
            call space(Npeng,Ng,Ts,CN,Vht,Vaux,Vtk,VD,Atr,Ata,Adr,Ada)
            bal=Ata-Atr
! Initial stability.
            call stabil(KG,Cgmb)
 1     format(2x,a,t45,es10.3e2,x,a)
 2     format(2x,a,t45,f6.2,x,a)
 3     format(2x,a)
 5     format(2x,a,t45,f5.3)
 6     format(2x,a,t45,f5.2)
       return
            end

            subroutine resist(Ve,SHPe,Vs,Vmax)
! This subroutine calculates the sustained and maximal speed of the ship,
! as well as the endurance SHP. It uses additional functions, also.
      use msimsl
            real LWL,sustain,speedmax
            external sustain,speedmax
            common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
            common/dimen/LWL,Vfl
            common/wet/S
            A0=7.018-2.331*Cbt+0.199*Cbt**2
            A1=-11.0+5.536*Cbt-0.704*Cbt**2
            A2=6.913-3.419*Cbt+0.451*Cbt**2
            Cstss=A0+A1*Cp+A2*Cp**2
            Stss=Cstss*sqrt(Vfl*LWL)
            Ss=Stss
            Ssd=1400.0
            S=Ss+Ssd
            write(10,*)
            write(10,1) 'Speed and Resistance ...'
            write(10,*)
            write(10,2) 'Wetted area',s,'[ft 2]'
! Calculate sustained speed.
            v1=10.0
            v2=40.0
            nmax=100
            call zbren(sustain,10.0,0.001,v1,v2,nmax)
            Vs=v2
            write(10,3) 'Sustained speed',Vs,'[knt]'
! Calculate maximal speed.
            v1=10.0
            v2=40.0
            nmax=100
            call zbren(speedmax,10.0,0.001,v1,v2,nmax)
            Vmax=v2
            write(10,3) 'Maximal speed',Vmax,'[knt]'
            SHPe=shp(Ve)
            write(10,2) 'Endurance shaft horsepower',SHPe,'[hp]'
 1     format(2x,a)
 2     format(2x,a,t45,f7.1,x,a)
 3     format(2x,a,t45,f5.2,x,a)
            return
            end

            real function sustain(v)
! This function defines the required power balance for sustained speed condition.
            common/propuls/Npeng,Pbpeng
            sustain=1.25*shp(v)/0.97-Npeng*Pbpeng
            return
            end
```

```
          real function speedmax(v)
! This function defines the required power balance for maximal speed condition.
          common/propuls/Npeng,Pbpeng
          speedmax=shp(v)/0.97-Npeng*Pbpeng
          return
          end

          real function shp(U)
! This function calculates shaft horsepower of the ship at a given speed.
          real LWL,WCF1(31),WCF2(31),interp
          common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
          common/dimen/LWL,Vfl
          common/dimen1/T,B,HDK,D10
          common/propuls/Npeng,Pbpeng
          common/wet/S
          common/prop/Dp
          PMF=1.08
          Nfins=0
          ro=1.9905              ! Sea water density in [lbf*s^2/ft^4].
          R=U/sqrt(LWL)          ! V here in knots, because Gertler tables are in [knt].
          V=U*1.69               ! Convert to [ft/sec].
! Frictional resistance.
          RN=LWL*V/1.2817e-5
          CF=0.075/(log10(RN)-2)**2
          RF=0.5*ro*S*(CF+0.0004)*V**2
! Residuary resistance.
          call resid(R,CRTSS)
          RRTSS=0.5*ro*S*CRTSS*V**2/1000
          open(15,file='c:\thesis\analysis\Norm_hull.prn',status='old')
          do 1 i=1,31
                read(15,*) WCF1(i),WCF2(i)
 1     continue
          close(15)
          j=0
          do while (R.ge.WCF1(j+1))
                j=j+1
          end do
          WCF=interp(WCF2(j),WCF2(j+1),WCF1(j),WCF1(j+1),R)
          RR=RRTSS*WCF
! Bare hull total resistance.
          RT=RR+RF
! Effective horse power.
          PEBH=RT*V/550                  ! Bare hull, converted to [hp].
          CDAPP=(-4e-9*LWL**3+9e-6*LWL**2-0.0081*LWL+5.0717)*1e-5/1.69**3   ! in
[hp*sec^3/ft^5].
          Cprop=1.0
          Dp=(0.64*T+0.013*LWL)*Cprop
          PEfins=0.025*PEBH
          if(Nfins.eq.0) PEfins=0.0
          PEAPP=1.23*LWL*Dp*CDAPP*V**3+PEfins ! Appendages, in [hp].
          Av=1.05*B*(D10-T+3*HDK)
          PEAA=0.35*Av*0.0023817*V**3/550     ! Air drag, in [hp].
          PET=PEBH+PEAPP+PEAA            ! Total effective power, in [hp].
          EHP=PET*PMF
          shp=EHP/0.67           ! Shaft horsepower.
       return
          end

          subroutine resid(R,CRTSS)
! This subroutine calculates Gertler residuary drag coefficient.
       real LWL,c(38,16),interp
          character*29 file1,file2
          common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
          common/dimen/LWL,Vfl
          Cv=Vfl/LWL**3
          k=1
          if(Cv.lt.0.001) Cv=0.001
          if(Cv.gt.0.006) Cv=0.006
          do while (Cv.ge.(k+1)*0.001)
                k=k+1
          end do
          select case(k)
                case(1)
                      file1='c:\thesis\analysis\Cv1.prn'
                      file2='c:\thesis\analysis\Cv2.prn'
```

```
              case(2)
                     file1='c:\thesis\analysis\Cv2.prn'
                     file2='c:\thesis\analysis\Cv3.prn'
              case(3)
                     file1='c:\thesis\analysis\Cv3.prn'
                     file2='c:\thesis\analysis\Cv4.prn'
              case(4)
                     file1='c:\thesis\analysis\Cv4.prn'
                     file2='c:\thesis\analysis\Cv5.prn'
              case(5)
                     file1='c:\thesis\analysis\Cv5.prn'
                     file2='c:\thesis\analysis\Cv6.prn'
              case(6)
                     file1='c:\thesis\analysis\Cv6.prn'
                     file2='c:\thesis\analysis\Cv6.prn'
         end select
         open(15,file=file1,status='old')
         do 1 i=1,36
              read(15,*) (c(i,j),j=1,16)
1     continue
      close(15)
      m=0
         if(R.lt.0.5) R=0.5
         if(R.gt.2.0) R=2.0
      do while (R.ge.0.5+0.1*m)
              m=m+1
         end do
         n=0
      do while (Cp.ge.0.48+0.02*n)
              n=n+1
         end do
         x1=interp(c(n,m),c(n+1,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         x2=interp(c(n,m+1),c(n+1,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         c1R225=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
         x1=interp(c(n+12,m),c(n+13,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         x2=interp(c(n+12,m+1),c(n+13,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         c1R300=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
         x1=interp(c(n+24,m),c(n+25,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         x2=interp(c(n+24,m+1),c(n+25,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         c1R375=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
         open(15,file=file2,status='old')
         do 2 i=1,36
              read(15,*) (c(i,j),j=1,16)
2     continue
      close(15)
         x1=interp(c(n,m),c(n+1,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         x2=interp(c(n,m+1),c(n+1,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         c2R225=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
         x1=interp(c(n+12,m),c(n+13,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         x2=interp(c(n+12,m+1),c(n+13,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         c2R300=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
         x1=interp(c(n+24,m),c(n+25,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         x2=interp(c(n+24,m+1),c(n+25,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
         c2R375=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
         CR225=interp(c1R225,c2R225,k*0.001,(k+1)*0.001,Cv)
         CR300=interp(c1R300,c2R300,k*0.001,(k+1)*0.001,Cv)
         CR375=interp(c1R375,c2R375,k*0.001,(k+1)*0.001,Cv)
         FF=4./3*(Cbt-3)
         CRTSS=CR300+FF*(CR375-CR225)/2+FF**2*((CR375+CR225)/2-CR300)
         return
         end

         real function interp(y1,y2,x1,x2,x)
! This function performes linear interpolation.
         interp=y1+(y2-y1)*(x-x1)/(x2-x1)
         return
         end

         subroutine availspac(Cp,Crd,Vht,CH,Dav)
! This subroutine calculates the available space in the ship.
         real LWL
         common/dimen/LWL,Vfl
         common/dimen1/T,B,HDK,D10
         common/calsea/Hr
         Hr=9.5
```

```
          D0=2.011827*T-6.36215e-6*LWL**2+2.780649e-2*LWL
          D20=0.014*LWL*(2.125+1.25e-3*LWL)+T
          F0=D0-T
          F10=D10-T
          F20=D20-T
          Apro=LWL/0.98*(F0+4*F10+F20)/6
          Fav=Apro/LWL
          Dav=Fav+T
          CN=LWL*B*Dav/1e5
          Cw=0.278+0.836*Cp
          Vhaw=Fav**2*tand(10.0)*LWL+LWL*B*Cw*Fav
          Bmax=B+2*tand(10.0)*Fav
          Blow=Bmax-2*tand(10.0)*Hr
          Vhl=Crd*LWL*(Bmax+Blow)/2*Hr*Cw
          Vht=Vfl+Vhaw-Vhl
          write(10,*)
          write(10,1) 'Space Available ...'
          write(10,*)
          write(10,2) 'D0',D0,'[ft]'
          write(10,2) 'D10',D10,'[ft]'
          write(10,2) 'D20',D20,'[ft]'
          write(10,3) 'Total hull volume',Vht,'[ft^3]'
 1      format(2x,a)
 2      format(2x,a,t45,f5.2,x,a)
 3      format(2x,a,t45,f9.1,x,a)
        return
          end

          subroutine electric(Vht,VD,PI,Wcps,KW24avg,Vaux,KWgreq)
! This subroutine calculates electrical load and auxiliary machinery rooms
! total volume.All loads in [KW].
        real LWL,KWp,KWs,KWe,KWm,KWcps,KWb,KWf,KWhn,KWa,KWserv,KWnp,KWpay
          real KWmfl,KWh,KWv,KWac,KWmflm,KWgreq,KW24,KW24avg
          common/dimen/LWL,Vfl
          common/dimen1/T,B,HDK,D10
          common/man/NO,NE,NA
          common/elect/Ng
          EDMF=1.0
          EFMF=1.01
          E24MF=1.2
          KWpay=947.9
          KWp=0.00323*PI
          KWs=0.00826*LWL*T
          VT=Vht+VD
          KWe=0.000213*VT
          KWm=101.4
          KWcps=0.0
          if(Wcps.gt.0.0) KWcps=0.000135*VT
          NT=NO+NE
          KWb=0.235*NT
          KWf=0.000097*VT
          KWhn=0.000177*Vht
          KWa=0.65*NT
          KWserv=0.395*NT
          KWnp=KWp+KWs+KWe+KWm+KWb+KWf+KWhn+KWa+KWserv
! Iterative loop for net electrical load and AMR volume.
        Vmb=138620.0              ! MMR volume.
          KWmfl=3000.0            ! First guess.
 1        Vaux=56900.0*KWmfl/3411.0
          KWh=0.00064*(VT-Vmb-Vaux)
          KWv=0.103*(KWh+KWpay)+KWcps
          KWac=0.67*(0.1*NT+0.00067*(VT-Vmb-Vaux)+0.1*KWpay)
          f=KWnp+KWh+KWv+KWac+KWpay
          if(abs((KWmfl-f)/KWmfl).gt.0.01) then
                KWmfl=f
                goto 1
          endif
          KWmflm=EDMF*EFMF*KWmfl
          KWgreq=KWmflm/(Ng-1)/0.9
          KW24=0.5*(KWmfl-KWp-KWs)+KWp+KWs
          KW24avg=E24MF*KW24
          write(10,*)
          write(10,1) 'Electrical Loads ...'
          write(10,*)
          write(10,3) 'Winter cruise electrical load',KWmfl,'[KW]'
        write(10,3) 'Marginal winter cruise electrical load',KWmflm,'[kW]'
        write(10,3) '24 hours average electrical load',KW24avg,'[kW]'
        write(10,3) 'Power required per generator',KWgreq,'[kW]'
        write(10,4) 'Auxiliary machinery rooms volume',Vaux,'[ft 3]'
2     format(2x,a)
3     format(2x,a,t45,f6.1,x,a)
4     format(2x,a,t45,f7.1,x,a)
      return
        end

        subroutine tank(E,Ve,Vmax,SHPe,KW24avg,Pi,Vtk)
! This subroutine calculates fuel weight and volume of all tanks.
      common/liquid/WF46,WF52,WF41
        common/man/NO,NE,NA
        real KW24avg
        WF42=64.4
        NT=NO+NE
        Pebavg=1.1*SHPe/0.97
        rppe=Pebavg/(Pi/2.0)
      rnpe=Ve/Vmax
        rqpe=rppe/rnpe
        SFCpe=0.4097          ! [lb/hp*hr].
        SFCepe=SFCpe/rppe*(7.215e-2*exp(1.252*rnpe)+0.3629*rqpe*          &
          exp(0.7229*rnpe))
        f1=1.02
        if(1.1*SHPe.le.Pi/6) f1=1.04
        if(1.1*SHPe.ge.Pi/3) f1=1.03
        FRsp=f1*SFCepe
        FRavg=1.05*FRsp
        Wbp=E/Ve*Pebavg*FRavg/2240.0                     ! [ltons].
        Pg24=KW24avg/(0.7457*0.961*0.989)
        rpg=Pg24/9200.0
        rng=1.0
        rqg=rpg/rng
        SFCg=0.4727           ! [lb/hp*hr].
        SFCg24=SFCg/rpg*(0.2821+0.7179*rqg)
        SFCge24=SFCg24*Pg24/KW24avg
        FRgsp=1.04*SFCge24
        FRgavg=1.05*FRgsp
      Wbe=E/Ve*KW24avg*FRgavg/2240.0
        WF41=(Wbp+Wbe)/0.95
        Vf=1.02*1.05*42.3*WF41
        Vhf=1.02*1.05*43.0*WF42
        WF46=17.6
        Vlo=1.02*1.05*39.9*WF46
        WF52=NT*0.15
        Vw=1.02*36*WF52
        Vsew=(NT+NA)*2.005
        Vwaste=0.02*Vf
        Vbal=0.19*Vf
        Vtk=Vf+Vhf+Vlo+Vw+Vsew+Vwaste+Vbal
        write(10,*)
        write(10,1) 'Tankage ...'
        write(10,*)
        write(10,2) 'Fuel weight',WF41,'[lton]'
        write(10,3) 'Fuel tanks total volume',Vf,'[ft 3]'
        write(10,3) 'Ballast tank volume',Vbal,'[ft 3]'
        write(10,3) 'Total tanks volume',Vtk,'[ft 3]'
1     format(2x,a)
2     format(2x,a,t45,f6.1,x,a)
3     format(2x,a,t45,f7.1,x,a)
      return
        end

        subroutine weight(Wfl,Pi,Vt,LWL,CN,Ts,Ng,KWg,VD,Dav,KG,NHPIE,     &
          ERRw,Hmb)
! This subroutine calculates the weight (full load and light weight).
      real LWL,KWg,KGmarg,KG
        common/weig/Wops
        common/liquid/WF46,WF52,WF41
        common/prop/Dp
        common/man/NO,NE,NA
        common/dimen1/T,B,HDK,D10
        WMF=0.005
        KGmarg=0.5
```

```
Wvp=334.8
VCGvp=23.7
Wp100=81.2
VCGp100=40.1
Wp400=176.5
VCGp400=45.5
Wp500=15.7
VCGp500=32.2
Wp600=0.0
VCGp600=0.0
Wp700=314.6
VCGp700=35.3
W498=86.5
VCG498=-3.95
W237=0.0
VCG237=0.0
Wbm=P1*(9.0+12.4*(P1*1e-5-1)**2)/2240.0
Ws=0.41*LWL
Wpr=0.174*Dp**(5.497-0.0433*Dp)/2240.0
Wb=0.235*(Ws+Wpr)
Wst=Ws+Wb+Wpr
W2=Wbm+Wst+W237
W3=50.0+0.0352*Ng*KWg
Wic=4.45e-5*Vt
Wcc=2.2*CN
Woc=0.4*(Wp400+Wic+Wcc)
W4=Wp400+Wic+Wcc+Woc+W498
W598=62e-6*Vt
NT=NO+NE
Waux=(0.000772*Vt**1.443+5.14*Vt+6.19*Vt**0.7224+377.*NT+2.74*P1)&
  *1e-4+117.0
W593=10.0
W5=Waux+Wp500+W593+W598+Wops
Wofh=4.18e-4*Vt
Wofp=0.8*(NT-9.5)
W6=Wofh+Wofp+Wp600
W7=Wp700
Wbh=0.93*1.135*(1.68341*CN**2+167.1721*CN-103.293)
Wdh=0.00168*VD
W171=2.0
W180=0.0735*(Wbh+W2+W3+W4+W5+W6+W7)
W1=Wbh+Wdh+W171+W180+Wp100
Wm24=WMF*(W1+W2+W3+W4+W5+W6+W7)
W1s=W1+W2+W3+W4+W5+W6+W7+Wm24
WF31=NT*2.45e-3*Ts
WF32=0.00071*Ts*NT+0.0049*NT
WF10=(236*NE+400*(NO+1))/2240.0
Wt=W1s+Wvp+WF41+WF46+WF52+WF31+WF32+WF10
ERRw=abs((Wt-Wf1)/Wt)
if(ERRw.gt.0.01) then
      Wf1=Wt
      return
endif
VCGbh=0.51*D10
VCGdh=D10+1.5*HDK
VCG180=0.5*D10
VCG171=D10+0.13*LWL
P100=Wbh*VCGbh+Wdh*VCGdh+W180*VCG180+W171*VCG171+Wp100*VCGp100
VCGbm=0.5*D10
VCGst=4.8+0.35*T
P200=Wbm*VCGbm+Wst*VCGst+W237*VCG237
VCG300=0.55*D10
P300=W3*VCG300
VCGic=D10
VCGcc=5.6+0.4625*D10
VCGoc=0.5*D10
P400=Wic*VCGic+Wcc*VCGcc+Woc*VCGoc+W498*VCG498+Wp400*VCGp400
VCGaux=0.9*(D10-9.4)
P500=Waux*VCGaux+Wp500*VCGp500
VCGofh=0.65*D10
VCGofp=4.2+0.4*D10
P600=Wofh*VCGofh+Wofp*VCGofp+Wp600*VCGp600
P700=Wp700*VCGp700
Pwg=P100+P200+P300+P400+P500+P600+P700
VCG1s=Pwg/(W1s-Wm24)
```

```
         VCGF10=0.732*D10
         VCGF31=0.523*Dav
         VCGF32=0.592*Dav
         VCGF41=10.3
         Hmb=D10-NHPIE*HDK/2
         VCGF46=0.53*Hmb
         VCGF52=0.138*Dav
         Pwg1=WF10*VCGF10+WF31*VCGF31+WF32*VCGF32+WF41*VCGF41+WF46*VCGF46+&
           WF52*VCGF52+Wvp*VCGvp
         KG=(Pwg+Pwg1)/Wt+KGmarg
      write(10,*)
         write(10,1) 'Weight and Center of gravity ...'
         write(10,*)
         write(10,2) 'Lightweight',Wls,'[lton]'
         write(10,3) 'Vertical CG of lightship',VCGls,'[ft]'
         write(10,2) 'Full load displacement',Wt,'[lton]'
         write(10,3) 'Vertical CG at full load',KG,'[ft]'
 1    format(2x,a)
 2    format(2x,a,t45,f7.1,x,a)
 3    format(2x,a,t45,f5.2,x,a)
      return
         end

         subroutine space(Npeng,Ng,Ts,CN,Vht,Vaux,Vtk,VD,Atr,Ata,Adr,Ada)
! This subroutine calculates space requirements.
      common/man/NO,NE,NA
         common/dimen1/T,B,HDK,D10
         common/duct/NDIE,NHPIE,NHeIE
         ADPA=579.0
         ADPC=3464.0
         AHPA=5173.0
         AHPC=6292.0
         AIE=159.1
         AGIE=35.7
         Adpr=1.15*ADPA+1.23*ADPC
         Ahpr=1.15*AHPA+1.23*AHPC
         Acono=225.0
         Ado=75.0*NO
         Adl=Acono+Ado
         Ahab=50.0
         NT=NO+NE
         Ahl=Ahab*(NT+NA)-Adl
         Ahs=300.0+0.0158*NT*9*Ts
         Adm=0.05*(ADPA-Adl)
         Adb=16*(B-13.0)
         Ahsf=1750.0*CN
         Apie=Npeng*AIE
         Aeie=Ng*AGIE
         Adie=1.4*NDIE*(Apie+Aeie)
         Ahie=1.4*(NHPIE*Apie+NHeIE*Aeie)
         Ahr=Ahpr+Ahl+Ahs+Ahsf+Ahie
         Vhr=HDK*Ahr
         Adr=Adpr+Adl+Adm+Adb+Adie
         Vdr=HDK*Adr
         Atr=Ahr+Adr
         Vtr=Vhr+Vdr
         Vmb=138620.0
         Vha=Vht-Vmb-Vaux-Vtk
         Aha=Vha/HDK
         Vta=Vha+VD
         Ada=VD/HDK
         Ata=Aha+Ada
         write(10,*)
         write(10,1) 'Area/Volume Balance ...'
         write(10,*)
         write(10,2) 'Required hull area',Ahr,'[ft^2]'
         write(10,2) 'Available hull area',Aha,'[ft^2]'
         write(10,3) 'Required hull volume',Vhr,'[ft^3]'
         write(10,3) 'Available hull volume',Vha,'[ft^3]'
         write(10,2) 'Required deckhouse area',Adr,'[ft^2]'
         write(10,2) 'Available deckhouse area',Ada,'[ft^2]'
         write(10,3) 'Required deckhouse volume',Vdr,'[ft^3]'
         write(10,3) 'Available deckhouse volume',VD,'[ft^3]'
         write(10,2) 'Total required area',Atr,'[ft^2]'
         write(10,2) 'Total available area',Ata,'[ft^2]'
```

```
          write(10,3) 'Total required volume',Vtr,'[ft^3]'
          write(10,3) 'Total available volume',Vta,'[ft^3]'
 1      format(2x,a)
 2      format(2x,a,t45,f7.1,x,a)
 3      format(2x,a,t45,f8.1,x,a)
        return
          end

          subroutine stabil(KG,Cgmb)
! This subroutine calculates initial stability parameters.
        real KG,LWL,KB
          common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
          common/dimen1/T,B,HDK,D10
          common/dimen/LWL,Vfl
          Cw=0.278+0.836*Cp
          Cit=-0.537+1.44*Cw
          KB=T/3*(2.4-Cp*Cx/Cw)
          BM=LWL*B**3*Cit/12/Vfl
          GM=KB+BM-KG
          Cgmb=GM/B
          write(10,*)
          write(10,1) 'Initial Stability .....'
          write(10,*)
          write(10,2) 'KB',KB,'[ft]'
          write(10,2) 'BM',BM,'[ft]'
          write(10,2) 'Metacentric height',GM,'[ft]'
          write(10,3) 'GM to B ratio',Cgmb
 1      format(2x,a)
 2      format(2x,a,t45,f5.2,x,a)
 3      format(2x,a,t45,f5.3)
        return
          end

          subroutine initial(pop_size)
! This subroutine creates the initial population for the search.
        real C_p(30),C_x(30),C_dl(30),C_bt(30),C_D10(30),C_rd(30)
          integer pop_size
          common/genes/C_p,C_x,C_dl,C_bt,C_D10,C_rd
          common/limits/Cpmin,Cpmax,Cxmin,Cxmax,Cdlmin,Cdlmax,Cbtmin,         &
            Cbtmax,CD10min,CD10max,Crdmin,Crdmax
          do 1 i=1,pop_size
                C_p(i)=gene(Cpmin,Cpmax)
                C_x(i)=gene(Cxmin,Cxmax)
                C_rd(i)=gene(Crdmin,Crdmax)
                C_dl(i)=gene(Cdlmin,Cdlmax)
                C_D10(i)=gene(CD10min,CD10max)
                C_bt(i)=gene(Cbtmin,Cbtmax)
 1      continue
          return
          end

          real function gene(Cmin,Cmax)
! This function randomly calculates values for genes.
          real ran(1)
          call rnun(1,ran)
          gene=Cmin+(Cmax-Cmin)*ran(1)
          return
          end

          subroutine select_class(F,pop_size,gen)
! This subroutine constructs roulette wheel based on classical approach.
        real F_p(30),p(30),q(30)
          integer pop_size,gen
          common/fitness/F_p
          common/prob/p,q
          open(20,file='c:\thesis\analysis\DDG_select.out',status='replace')        ! File
containing selection report.
          write(20,101) 'Classical selection, after evaluation of generati&
            on #',gen
          write(20,*)
          write(20,102) 'i','F_p(i)','p(i)','q(i)','r(i)','i_sel(i)'
          write(20,*)
! Creation of the roulette wheel.
          do 1 i=1,pop_size
                p(i)=F_p(i)/F
```

```
              q(i)=0.0
              do 2 j=1,i
                     q(i)=q(i)+p(j)
 2            continue
 1       continue
! 'Rotation' of the wheel, pop_size times.
         call rotation(pop_size)
 101  format(2x,a,i2)
 102  format(t4,a,t8,a,t17,a,t25,a,t33,a,t39,a)
         return
         end

         subroutine select_rank(pop_size,gen,select_method,press_method)
! This subroutine constructs the roulette wheel based on rank.
         use msimsl
         real F_p(30),dummy1(30),p(30),q(30)
         integer pop_size,gen,k(30)
         character*1 select_method,press_method
         common/fitness/F_p
         common/prob/p,q
         if(press_method.eq.'n') qq=-0.06*(gen/100.0)+0.98
         if(press_method.eq.'u') qq=0.94
         a1=(1.0-qq)/(1.0-qq**pop_size)      ! First term in series; designed as to get a
total sum of 1.0.
         open(20,file='c:\thesis\analysis\DDG_select.out',status='replace')        ! File
containing selection report.
         write(20,101) 'Rank based selection, after evaluation of generati
           on #',gen
         write(20,103) 'Selection pressure parameter ...',qq
         write(20,*)
         write(20,102) 'i','F_p(i)','p(i)','q(i)','r(i)','i_sel(i)'
         write(20,*)
         do 1 i=1,pop_size    ! Initialization of vector k.
              k(i)=i
 1    continue
         call svrgp(pop_size,-F_p,dummy1,k)         ! Minus sign to get descending order.
! Creation of the roulette wheel.
         do 2 i=1,pop_size
              p(k(i))=a1*qq**(i-1)
 2    continue
         do 3 i=1,pop_size
              q(i)=0.0
              do 4 j=1,i
                     q(i)=q(i)+p(j)
 4            continue
 3       continue
! 'Rotation' of the wheel, pop_size times.
         if(select_method.eq.'r') call rotation(pop_size)
         if(select_method.eq.'b') call rot_baker(pop_size)
 101  format(2x,a,i2)
 102  format(t4,a,t8,a,t17,a,t25,a,t33,a,t39,a)
 103  format(2x,a,x,f5.3)
         return
         end

         subroutine rotation(pop_size)
! This subroutine rotates the roulette wheel pop_size times and performs selection.
         real F_p(30),r(30),p(30),q(30)
         integer pop_size,i_sel(30)
         common/fitness/F_p
         common/prob/p,q
! 'Rotation' of the wheel, pop_size times.
         call rnun(pop_size,r)
         do 5 i=1,pop_size
              j=1
              if(r(i).le.q(1)) then
                     i_sel(i)=j
                     write(20,100) i,F_p(i),p(i),q(i),r(i),i_sel(i)
                     goto 5
              endif
              do 6 while(r(i).gt.q(j))
                     j=j+1
 6          continue
              i_sel(i)=j
              write(20,100) i,F_p(i),p(i),q(i),r(i),i_sel(i)
 5    continue
      write(20,*)
! Reproduction.
         call reproduct(pop_size,i_sel)
 100  format(2x,i2,2x,f6.4,2x,f6.4,2x,f6.4,2x,f5.3,6x,i2)
         return
         end

         subroutine rot_baker(pop_size)
! This subroutine rotates the roulette wheel one time, but with pop_size equaly
! spaced markers, according to Baker's selection method.
         real F_p(30),r(30),p(30),q(30)
         integer pop_size,i_sel(30)
         common/fitness/F_p
         common/prob/p,q
         delta=1.0/pop_size
! Construction of pop_size equaly spaced markers.
         call rnun(1,r)
         do 1 i=2,pop_size
               r(i)=r(i-1)+delta
               if(r(i).gt.1.0) r(i)=r(i)-1.0
 1       continue
! 'Rotation' of the wheel once.
         do 5 i=1,pop_size
               j=1
               if(r(i).le.q(1)) then
                       i_sel(i)=j
                       write(20,100) i,F_p(i),p(i),q(i),r(i),i_sel(i)
                       goto 5
               endif
               do 6 while(r(i).gt.q(j))
                       j=j+1
 6          continue
               i_sel(i)=j
               write(20,100) i,F_p(i),p(i),q(i),r(i),i_sel(i)
 5    continue
      write(20,*)
! Reproduction.
         call reproduct(pop_size,i_sel)
 100  format(2x,i2,2x,f6.4,2x,f6.4,2x,f6.4,2x,f5.3,6x,i2)
         return
         end

         subroutine reproduct(pop_size,i_sel)
! This subroutine reproducts chromosomes for next generation, based on roulette wheel
! spuns.
         real C_p(30),C_n(30),C_dl(30),C_bt(30),C_D10(30),C_rd(30),F_p(30)
         real dummy(30)
         integer pop_size,i_sel(30)
         common/genes/C_p,C_n,C_dl,C_bt,C_D10,C_rd
         common/fitness/F_p
         dummy=C_p
         do 7 i=1,pop_size
               C_p(i)=dummy(i_sel(i))
 7    continue
         dummy=C_n
         do 8 i=1,pop_size
               C_n(i)=dummy(i_sel(i))
 8    continue
         dummy=C_dl
         do 9 i=1,pop_size
               C_dl(i)=dummy(i_sel(i))
 9    continue
      dummy=C_bt
         do 10 i=1,pop_size
               C_bt(i)=dummy(i_sel(i))
 10   continue
      dummy=C_D10
         do 11 i=1,pop_size
               C_D10(i)=dummy(i_sel(i))
 11   continue
      dummy=C_rd
         do 12 i=1,pop_size
               C_rd(i)=dummy(i_sel(i))
 12   continue
```

```
          dummy=F_p
          do 13 i=1,pop_size
                F_p(i)=dummy(i_sel(i))
 13   continue
          return
          end

          subroutine crossover(pop_size,gen,nn,n_cross)
! This subroutine performes the crossover genetic operator, on the chromosomes
! selected for next generation.
! nn-number of splits when crossover is performed.
          real C_p(30),C_x(30),C_dl(30),C_bt(30),C_D10(30),C_rd(30),r(30)
          real c1(6),c2(6),c(6)
          integer pop_size,gen,cross(30),r_dis(15),r2(1)
          common/genes/C_p,C_x,C_dl,C_bt,C_D10,C_rd
          Pc=0.25       ! Probability of crossover.
          a=1.0
          if(nn.eq.0) a=-0.009*gen+1.0
          call rnun(pop_size,r)
          n_cross=0            ! Counter of crossovered chromosomes.
! Decision of crossovered chromosomes.
          do 1 i=1,pop_size
                if(r(i).le.Pc) then
                        n_cross=n_cross+1
                        cross(n_cross)=i
                endif
 1    continue
          if(2*n_cross.ne.2*int(n_cross/2)) then      ! In case of an uneven number of
selections.
                call rnun(1,r)
                if(r(1).gt.0.5) then
                        n_cross=n_cross-1
                else
                        n_cross=n_cross+1
 5                      call rnund(1,pop_size,r_dis)
                        do 4 i=1,n_cross-1
                                if(r_dis(1).eq.cross(i)) goto 5              ! Ensuring that
a different chromosome is selected.
 4            continue
                        cross(n_cross)=r_dis(1)
                endif
          endif
! Mating the selected (for crossover) chromosomes.
          if(n_cross.eq.0) return
          call rnund(n_cross,5,r_dis)
          do 2 i=1,n_cross,2
                c1(1)=C_p(cross(i))
                c1(2)=C_x(cross(i))
                c1(3)=C_dl(cross(i))
                c1(4)=C_bt(cross(i))
                c1(5)=C_D10(cross(i))
                c1(6)=C_rd(cross(i))
                c2(1)=C_p(cross(i+1))
                c2(2)=C_x(cross(i+1))
                c2(3)=C_dl(cross(i+1))
                c2(4)=C_bt(cross(i+1))
                c2(5)=C_D10(cross(i+1))
                c2(6)=C_rd(cross(i+1))
                if(nn.eq.2) then
                        if(r_dis(i).eq.5) then ! In this case the crossover is 1 point
anyway.
                                r2(1)=6
                                goto 8
                        endif
                        call rnund(1,5-r_dis(i),r2)
                        r2(1)=r_dis(i)+r2(1)
                else
                        r2(1)=6
                endif
 8              do 3 j=r_dis(i)+1,r2(1)
                        c(j)=c1(j)
                        c2(j)=a*c1(j)+(1-a)*c2(j)
                        c1(j)=a*c(j)+(1-a)*c(j)
 3              continue
            C_p(cross(i))=c1(1)
```

```
                C_x(cross(i))=c1(2)
                C_dl(cross(i))=c1(3)
                C_bt(cross(i))=c1(4)
                C_D10(cross(i))=c1(5)
                C_rd(cross(i))=c1(6)
                C_p(cross(i+1))=c2(1)
                C_x(cross(i+1))=c2(2)
                C_dl(cross(i+1))=c2(3)
                C_bt(cross(i+1))=c2(4)
                C_D10(cross(i+1))=c2(5)
                C_rd(cross(i+1))=c2(6)
 2        continue
          return
          end

          subroutine mutation(mut_type,pop_size,Pm,n_mutat)
! This subroutine mutates randomly selected genes.
          real C_p(30),C_x(30),C_dl(30),C_bt(30),C_D10(30),C_rd(30),r(6)
          integer pop_size
          character*1 mut_type
          common/genes/C_p,C_x,C_dl,C_bt,C_D10,C_rd
          common/limits/Cpmin,Cpmax,Cxmin,Cxmax,Cdlmin,Cdlmax,Cbtmin,          &
            Cbtmax,CD10min,CD10max,Crdmin,Crdmax
          n_mutat=0
          do 1 i=1,pop_size
                call rnun(6,r)
                if(minval(r).le.Pm) n_mutat=n_mutat+1
                if(mut_type.eq.'u') then
                        if(r(1).le.Pm) C_p(i)=gene(Cpmin,Cpmax)
                        if(r(2).le.Pm) C_x(i)=gene(Cxmin,Cxmax)
                        if(r(3).le.Pm) C_rd(i)=gene(Crdmin,Crdmax)
                        if(r(4).le.Pm) C_dl(i)=gene(Cdlmin,Cdlmax)
                        if(r(5).le.Pm) C_D10(i)=gene(CD10min,CD10max)
                        if(r(6).le.Pm) C_bt(i)=gene(Cbtmin,Cbtmax)
                else
                        if(r(1).le.Pm) C_p(i)=C_p(i)+delta(Cpmin,Cpmax,C_p(i))
                        if(r(2).le.Pm) C_x(i)=C_x(i)+delta(Cxmin,Cxmax,C_x(i))
                        if(r(3).le.Pm) C_rd(i)=C_rd(i)+delta(Crdmin,Crdmax,C_rd(i))
                        if(r(4).le.Pm) C_dl(i)=C_dl(i)+delta(Cdlmin,Cdlmax,C_dl(i))
                        if(r(5).le.Pm) C_D10(i)=C_D10(i)+delta(CD10min,CD10max,     &
                          C_D10(i))
                        if(r(6).le.Pm) C_bt(i)=C_bt(i)+delta(Cbtmin,Cbtmax,C_bt(i))
                endif
 1    continue
      return
          end

          real function delta(Cmin,Cmax,C)
! This function performes non uniform mutation values.
          real r(1)
          integer r_dis(1),b,gen,gen_max
          common/gener/gen,gen_max
          b=2
          call rnund(1,2,r_dis)
          call rnun(1,r)
          if(r_dis(1).eq.1) then
                delta=-(C-Cmin)*(1-(r(1))**((1-gen/gen_max)**b))
          else
                delta=(Cmax-C)*(1-(r(1))**((1-gen/gen_max)**b))
          endif
          return
          end
```

Appendix G

DDGx Final Local Optimization Fortran Model

After the overall optimal hull is obtained from the evolutionary program of Appendix F, an additional local optimization, near the optimal hull, is performed using the hill climbing technique. The Fortran code written for that purpose uses the same self-balanced ship design model of Appendix F, since they both use the same fitness function. Figure G.1 displays the flowchart of the code.

The start point of the exploration is the best chromosome found using the evolutionary strategies. The algorithm then randomly picks one gene, and changes it randomly by plus or minus its required precision,

$$gene' = gene \pm \Delta gene \qquad \text{(G.1)}$$

where *Δgene* is the required precision, described in Table 5.4. If, for instance, the prismatic coefficient gene is selected, its new value may be C_P+0.001 or C_P-0.001. The new chromosome is then evaluated for the payload fraction using the same ship engineering model. If the new hull is feasible and lighter than the original one, it replaces it and the process is repeated. In case that the new vessel is non-feasible, or its payload fraction is lower, the process is repeated with a different selected gene. The number of iterations is limited to 50. The code also counts the number of times an improvement is obtained.

The analogy to hill climbing is clear: the computer walks in many directions in an effort to find a higher position on the hill. In this approach in which the climber does not view the whole hill the exploration is for a local maxima. However, it is believed that the payload fraction function is continuous, and that the starting point extracted from the genetic search ensures that this is an overall maxima.

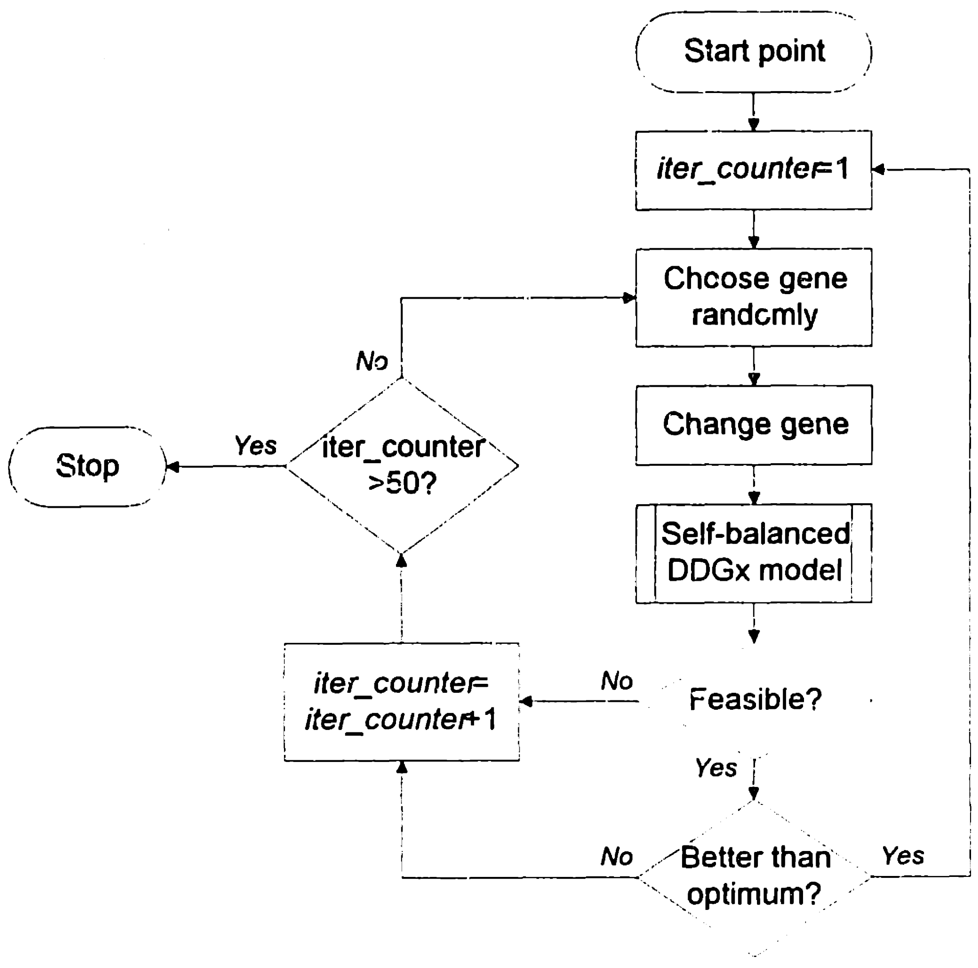


Figure G.1. *Hill climbing Fortran code flowchart.*

An example of an output from the program is shown in Figure G.2. The results of the first run of Table 5.5 are shown. The program writes the chromosome of each solution that provides a higher payload fraction. The small climbing steps create small improvements each time. Cases for which significant improvements occurr are rare.

```
#    Cp   Cx   Cdl   Cbt   CD10   Crd    Fp
 0  .70  .90  90.0  3.08  12.99  .00  .12529
 1  .70  .90  90.0  3.08  13.00  .00  .12531
 2  .70  .90  90.0  3.08  13.00  .00  .12534
 3  .70  .90  90.0  3.07  13.00  .00  .12537
 4  .70  .90  90.0  3.07  13.01  .00  .12538
 5  .70  .90  90.0  3.07  13.01  .00  .12542
 6  .70  .90  90.0  3.07  13.02  .00  .12543
 7  .70  .90  90.0  3.07  13.03  .00  .12544
 8  .70  .90  90.0  3.07  13.04  .00  .12545
 9  .70  .90  90.0  3.07  13.05  .00  .12547
10  .70  .90  90.0  3.07  13.06  .00  .12548
11  .70  .90  90.0  3.07  13.07  .00  .12549
12  .70  .90  90.0  3.07  13.08  .00  .12550
13  .70  .90  90.0  3.07  13.09  .00  .12551
14  .70  .90  90.0  3.07  13.10  .00  .12552
```

```
.
15  .70  .90  90.0  3.07  13.11  .00  .12554
16  .70  .90  90.0  3.07  13.12  .00  .12555
17  .70  .90  90.0  3.06  13.12  .00  .12558
18  .70  .90  90.0  3.06  13.13  .00  .12559
19  .70  .90  90.0  3.06  13.14  .00  .12560
20  .70  .90  90.0  3.06  13.15  .00  .12561
21  .70  .90  90.0  3.06  13.16  .00  .12562
22  .70  .90  90.0  3.06  13.17  .00  .12563
23  .70  .90  90.0  3.06  13.18  .00  .12564
24  .70  .90  90.0  3.05  13.18  .00  .12567
25  .70  .90  90.0  3.05  13.19  .00  .12569
26  .70  .90  90.0  3.05  13.20  .00  .12570
27  .70  .90  90.0  3.05  13.21  .00  .12571
28  .70  .90  90.0  3.05  13.22  .00  .12572
29  .70  .90  90.0  3.05  13.23  .00  .12573
30  .70  .90  90.0  3.05  13.24  .00  .12574
31  .70  .90  90.0  3.05  13.25  .00  .12575
32  .70  .90  90.0  3.05  13.26  .00  .12576
33  .70  .90  90.0  3.05  13.27  .00  .12578
34  .70  .90  90.0  3.04  13.27  .00  .12580

  Best ever fitness ...             .12580
  Number of improvements ... 34
  Cp ...                     .700
  Cx ...                     .900
  Cdl ...                   90.00
  Cbt ...                   3.04
  CD10 ...                  13.27
  Crd ...                    .000
```

Figure G.2. *Example of output file of hill climbing optimization program.*

The listing of the source file appears in the following pages.

```
    program ddg_hill_climbing
! Performes hill climbing search process for ship design.
! For units refer to Mathcad model; they are in U.S. system.
      use msimsl
          use msflib
          real bal,Vguess(1),VDK(1),LWL,KWgreq,KWg,gene(6,2)
          integer info(1),trials_counter,r_dis(1)
          character*2 warning_weight
          external bal
          common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
          common/dimen1/T,B,HDK,D10
          common/dimen/LWL,Vfl
          common/balance/Adr,Ada,KWgreq,KWg,Vs,Cgmb,Fp,warning_weight
          common/raised/Hr
          ! Limits of explored solution space.
          Cpmin=0.5
          Cpmax=0.7
          Cxmin=0.7
          Cxmax=0.9
          Cdlmin=60.0
          Cdlmax=90.0
          Cbtmin=2.8
          Cbtmax=3.7
          CD10min=10.0
          CD10max=15.0
          Crdmin=0.0
          Crdmax=0.8
          open(3,file='c:\thesis\analysis\par.in',status='old')
          read(3,*) Cp_start
          read(3,*) Cx_start
          read(3,*) Cdl_start
          read(3,*) Cbt_start
          read(3,*) CD10_start
          read(3,*) Crd_start
          close(3)
          Fp_max=0.0
          improv=-1
          trials_counter=0
          open(18,file='c:\thesis\analysis\hill.out',status='replace')
          write(18,29) '#','Cp','Cx','Cdl','Cbt','CD10','Crd','Fp'
! genes required accuracy.
          gene(1,2)=0.001
          gene(2,2)=0.001
          gene(3,2)=0.01
          gene(4,2)=0.01
          gene(5,2)=0.01
          gene(6,2)=0.01
 105  if(trials_counter.gt.50) goto 110
      write(*,24) trials_counter,'trials.'
          gene(1,1)=Cp_start
          gene(2,1)=Cx_start
          gene(3,1)=Cdl_start
          gene(4,1)=Cbt_start
          gene(5,1)=CD10_start
          gene(6,1)=Crd_start
          call rnund(1,6,r_dis)
      gene(r_dis(1),1)=climbing(gene(r_dis(1),1),gene(r_dis(1),2))
          Cp=gene(1,1)
          Cx=gene(2,1)
          Cdl=gene(3,1)
          Cbt=gene(4,1)
          CD10=gene(5,1)
          Crd=gene(6,1)
! Check deviatio,n from solution space.
          if((Cp.lt.Cpmin).or.(Cp.gt.Cpmax).or.(Cx.lt.Cxmin).or.            &
          - (Cx.gt.Cxmax).or.(Cdl.lt.Cdlmin).or.(Cdl.gt.Cdlmax).or.         &
                (Cbt.lt.Cbtmin).or.(Cbt.gt.Cbtmax).or.(CD10.lt.CD10min).or.    &
                (CD10.gt.Cd10max).or.(Crd.lt.Crdmin).or.(Crd.gt.Crdmax)) then
                        trials_counter=trials_counter+1
                        goto 105
          endif
! Iterative solver for deckhouse volume.
      Vguess(1)=243600.0      ! Initial guess for deckhouse volume.
          call zreal(bal,10.0,0.01,0.1,0.1,1,20,Vguess,VDK,info)
          if((warning_weight.eq.'on').or.(info(1).gt.20)) then
```

```
                trials_counter=trials_counter+1
                goto 105
           endif
           write(10,*)
           VD=VDK(1)
           write(10,1) 'Deckhouse volume',VD,'[ft^3]'
! Checking balance of ship (other than weight and space).
           write(10,*)
           write(10,3) 'Balance/Feasibility Status ...'
           write(10,*)
           write(10,4) 'Category','Required/Minimal','Available'
           write(10,6) 'Deckhouse area [ft^3]',Adr,Ada
           write(10,8) 'Electric plant [kW]',KWgreq,KWg
           Vsmin=29.96
           write(10,5) 'Sustained speed',Vsmin,Vs
           Cgmbmin=0.09
           Cgmbmax=0.135
           write(10,10) 'Initial stability',Cgmbmin,Cgmb
           write(10,11) Cgmbmax
           Hmbmin=11.0
           D10min=max(0.21*B+T,LWL/15,Hmbmin+HDK)
           if(Crd.ge.0.5) D10min=max(0.21*B+T+Hr,LWL/15+Hr,Hmbmin+HDK)
           write(10,5) 'Depth [ft]',D10min,D10
           VDmax=(2*B-4*HDK/tand(90.0))*HDK*0.6*LWL*1.1
           write(10,7) 'Maximal deckhouse volume [ft^3]',VDmax,VD
           Fpmin=0.075
           write(10,10) 'Payload fraction',Fpmin,Fp
           close(10)
           if((Ada.ge.Adr).and.(KWg.ge.KWgreq).and.(Vs.ge.Vsmin).and.          &
                (Cgmb.ge.Cgmbmin).and.(Cgmb.le.Cgmbmax).and.                    &
                (D10.ge.D10min).and.(VD.le.VDmax)) then
                trials_counter=trials_counter+1
                if(Fp.gt.Fp_max) then
                        trials_counter=0
                        improv=improv+1
                        Fp_max=Fp
                        Cp_start=Cp
                        Cx_start=Cx
                        Cdl_start=Cdl
                        Cbt_start=Cbt
                        CD10_start=CD10
                        Crd_start=Crd
                        write(*,24) improv,'improvements.'
                        write(*,25) 'Best ever so far ... ',Fp_max
                        write(*,*)
                        write(19,30) improv,Cp,Cx,Cdl,Cbt,CD10,Crd,Fp
                endif
           else
                trials_counter=trials_counter+1
           endif
           goto 105
 110   call beepqq(1000,1000)
           write(19,*)
           write(19,15) 'Best ever fitness ...',Fp_max
           write(19,13) 'Number of improvements ...',improv
           write(19,16) 'Cp ...',Cp_start
           write(19,16) 'Cx ...',Cx_start
           write(19,17) 'Cdl ...',Cdl_start
           write(19,18) 'Cbt ...',Cbt_start
           write(19,17) 'CD10 ...',CD10_start
           write(19,16) 'Crd ...',Crd_start
 1     format(2x,a,t45,f9.1,x,a)
 3     format(2x,a)
 4     format(2x,a,t40,a,t60,a)
 5     format(2x,a,t45,f5.1,t61,f5.1)
 6     format(2x,a,t45,f7.1,t61,f7.1)
 7     format(2x,a,t45,f9.1,t61,f9.1)
 8     format(2x,a,t45,f6.1,t61,f6.1)
 10    format(2x,a,t45,f5.3,t60,f7.5)
 11    format(t45,f5.3)
 13    format(2x,a,i3)
 15    format(2x,a,t35,f7.5)
 16    format(2x,a,t25,f5.3)
 17    format(2x,a,t25,f5.2)
 18    format(2x,a,t25,f4.2)
 24   format(2x,i2,x,a)
 25   format(2x,a,x,f7.5)
 29   format(t2,a,t6,a,t11,a,t16,a,t22,a,t28,a,t34,a,t41,a)
 30   format(i2,t4,f4.2,t9,f4.2,t15,f4.1,t21,f4.2,t27,f5.2,t33,f4.2,    &
        t39,f7.5)
          stop
          end

          real function bal(VD)
! This function bears the whole model, but actualy is used to calculate the
! area balance error. At each entrance to this function the weight is being balanced,
! and the resulting area error is calculated.
          real LWL,KWg,KW24avg,KG,KWgreq
          character*2 warning_weight
          common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
          common/dimen/LWL,Vfl
          common/dimen1/T,B,HDK,D10
          common/propuls/Npeng,Pbpeng
          common/man/NO,NE,NA
          common/elect/Ng
          common/liquid/WF46,WF52,WF41
          common/weig/Wops
          common/duct/NDIE,NHPIE,NHeIE
          common/balance/Adr,Ada,KWgreq,KWg,Vs,Cgmb,Fp,warning_weight
          Wp=922.9
          Wops=30.0
          NO=26
          NE=315
          NA=36
          Npeng=4
          Pbpeng=26775.0
          Ng=3
          KWg=2500.0
          NDIE=2
          NHPIE=2
          NHeIE=5
! Design requirements.
      HDK=10.66
          Ve=20.0
          E=3907.6
          Ts=45.0
          iter_wei=0            ! Counter of weight iterations.
          Wfl=Wp/0.1
 100  iter_wei=iter_wei+1
      if(iter_wei.gt.15) then
              write(*,3) 'Failed to converge in weight.'
              bal=0.0
              warning_weight='on'
              return
          endif
          open(10,file='c:\thesis\analysis\DDG_hill.out',status='replace')
          write(10,3) 'Main Parameters ...'
          write(10,*)
          write(10,5) 'Cp',Cp
          write(10,5) 'Cx',Cx
          write(10,2) 'Cdl',Cdl,'[lton/ft 3]'
          write(10,5) 'Cbt',Cbt
          write(10,8) 'CD10',CD10
          write(10,5) 'Crd',Crd
! Main dimensions.
      Vfl=Wfl*34.98
          LWL=100*(Wfl/Cdl)**(1./3)
          B=sqrt(Cbt*Vfl/(Cp*Cx*LWL))
          T=Vfl/(Cp*Cx*LWL*B)
          D10=LWL/CD10
          write(10,*)
          write(10,3) 'Main Dimensions on Waterline ...'
          write(10,*)
          write(10,1) 'Underwater volume',Vfl,'[ft 3]'
          write(10,2) 'Length on waterline',LWL,'[ft]'
          write(10,2) 'Beam',B,'[ft]'
          write(10,2) 'Draft',T,'[ft]'
          write(10,2) 'D10',D10,'[ft]'
! Resistance computations.
          call resist(Ve,SHPe,Vs,Vmax)
 ! Available space calculations.
       call availspac(Cp,Crd,Vht,CN,Dav)
 ! Electrical load calculations.
            Pi=Npeng*Pbpeng
            call electric(Vht,VD,Pi,Wcps,KW24avg,Vaux,KWgreq)
 ! Tankage volumes.
       call tank(E,Ve,Vmax,SHPe,KW24avg,Pi,Vtk)
 ! Weight.
            Vt=Vht+VD
            call weight(Wfl,Pi,Vt,LWL,CN,Ts,Ng,KWg,VD,Dav,KG,NHPIE,ERRw,Hmb)
            if(ERRw.gt.0.01) then
                  close(10,status='delete')
                  goto 100
            endif
            Fp=Wp/Wfl
            write(10,5) 'Payload fraction',Fp
 ! Space requirements and balance.
            call space(Npeng,Ng,Ts,CN,Vht,Vaux,Vtk,VD,Atr,Ata,Adr,Ada)
            bal=Ata-Atr
 ! Initial stability.
            call stabil(KG,Cgmb)
 1     format(2x,a,t45,es10.3e2,x,a)
 2     format(2x,a,t45,f6.2,x,a)
 3     format(2x,a)
 5     format(2x,a,t45,f5.3)
 6     format(2x,a,t45,f5.2)
       return
            end

            subroutine resist(Ve,SHPe,Vs,Vmax)
 ! This subroutine calculates the sustained and maximal speed of the ship,
 ! as well as the endurance SHP. It uses additional functions, also.
       use msimsl
            real LWL,sustain,speedmax
            external sustain,speedmax
            common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
            common/dimen/LWL,Vfl
            common/wet/S
            A0=7.026-2.331*Cbt+0.299*Cbt**2
            A1=-11.0+5.536*Cbt-0.704*Cbt**2
            A2=6.913-3.419*Cbt+0.451*Cbt**2
            Cstss=A0+A1*Cp+A2*Cp**2
            Stss=Cstss*sqrt(Vfl*LWL)
            Ss=Stss
            Ssd=1400.0
            S=Ss+Ssd
            write(10,*)
            write(10,1) 'Speed and Resistance ...'
            write(10,*)
            write(10,2) 'Wetted area',s,'[ft 2]'
! Calculate sustained speed.
            v1=10.0
            v2=40.0
            nmax=100
            call zbren(sustain,10.0,0.001,v1,v2,nmax)
            Vs=v2
            write(10,3) 'Sustained speed',Vs,'[knt]'
! Calculate maximal speed.
            v1=10.0
            v2=40.0
            nmax=100
            call zbren(speedmax,10.0,0.001,v1,v2,nmax)
            Vmax=v2
            write(10,3) 'Maximal speed',Vmax,'[knt]'
            SHPe=shp(Ve)
            write(10,2) 'Endurance shaft horsepower',SHPe,'[hp]'
 1     format(2x,a)
 2     format(2x,a,t45,f7.1,x,a)
 3     format(2x,a,t45,f5.2,x,a)
            return
            end

            real function sustain(v)
! This function defines the required power balance for sustained speed condition.
          common/propuls/Npeng,Pbpeng
       sustain=1.25*shp(v)/0.97-Npeng*Pbpeng
       return
       end

       real function speedmax(v)
! This function defines the required power balance for maximal speed condition.
       common/propuls/Npeng,Pbpeng
       speedmax=shp(v)/0.97-Npeng*Pbpeng
       return
       end

       real function shp(U)
! This function calculates shaft horsepower of the ship at a given speed.
       real LWL,WCF1(31),WCF2(31),interp
       common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
       common/dimen/LWL,Vfl
       common/dimen1/T,B,HDK,D10
       common/propuls/Npeng,Pbpeng
       common/wet/S
       common/prop/Dp
       PMF=1.08
       Nfins=0
       ro=1.9905              ! Sea water density in [lbf*s^2/ft^4].
       R=U/sqrt(LWL)          ! V here in knots, because Gertler tables are in [knt].
       V=U*1.69               ! Convert to [ft/sec].
! Frictional resistance.
       RN=LWL*V/1.2817e-5
       CF=0.075/(log10(RN)-2)**2
       RF=0.5*ro*S*(CF+0.0004)*V**2
! Residuary resistance.
       call resid(R,CRTSS)
       RRTSS=0.5*ro*S*CRTSS*V**2/1000
       open(15,file='c:\thesis\analysis\Worm_hull.prn',status='old')
       do 1 i=1,31
            read(15,*) WCF1(i),WCF2(i)
 1    continue
       close(15)
       j=0
       do while (R.ge.WCF1(j+1))
            j=j+1
       end do
       WCF=interp(WCF2(j),WCF2(j+1),WCF1(j),WCF1(j+1),R)
       RR=RRTSS*WCF
! Bare hull total resistance.
       RT=RR+RF
! Effective horse power.
       PEBH=RT*V/550                  ! Bare hull, converted to [hp].
       CDAPP=(-4e-9*LWL**3+9e-6*LWL**2-0.0081*LWL+5.0717)*1e-5/1.69**3   ! in
[hp*sec^3/ft^5].
       Cprop=1.0
       Dp=(0.64*T+0.013*LWL)*Cprop
       PEfins=0.025*PEBH
       if(Nfins.eq.0) PEfins=0.0
       PEAPP=1.23*LWL*Dp*CDAPP*V**3+PEfins ! Appendages, in [hp].
       Aw=1.05*B*(D10-T+3*HDK)
       PEAA=0.35*Aw*0.0023817*V**3/550     ! Air drag, in [hp].
       PET=PEBH+PEAPP+PEAA            ! Total effective power, in [hp].
       EHP=PET*PMF
       shp=EHP/0.67           ! Shaft horsepower.
    return
       end

       subroutine resid(R,CRTSS)
! This subroutine calculates Gertler residuary drag coefficient.
    real LWL,c(36,16),interp
       character*29 file1,file2
       common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
       common/dimen/LWL,Vfl
       Cv=Vfl/LWL**3
       k=0
       if(Cv.lt.0.001) Cv=0.001
       if(Cv.gt.0.006) Cv=0.006
       do while (Cv.ge.(k+1)*0.001)
            k=k+1
       end do
```

```
          select case(k)
                case(1)
                        file1='c:\thesis\analysis\Cv1.prn'
                        file2='c:\thesis\analysis\Cv2.prn'
                case(2)
                        file1='c:\thesis\analysis\Cv2.prn'
                        file2='c:\thesis\analysis\Cv3.prn'
                case(3)
                        file1='c:\thesis\analysis\Cv3.prn'
                        file2='c:\thesis\analysis\Cv4.prn'
                case(4)
                        file1='c:\thesis\analysis\Cv4.prn'
                        file2='c:\thesis\analysis\Cv5.prn'
                case(5)
                        file1='c:\thesis\analysis\Cv5.prn'
                        file2='c:\thesis\analysis\Cv6.prn'
                case(6)
                        file1='c:\thesis\analysis\Cv6.prn'
                        file2='c:\thesis\analysis\Cv6.prn'
          end select
          open(15,file=file1,status='old')
          do 1 i=1,36
                read(15,*) (c(i,j),j=1,16)
1     continue
      close(15)
      m=0
          if(R.lt.0.5) R=0.5
          if(R.gt.2.0) R=2.0
      do while (R.ge.0.5+0.1*m)
                m=m+1
          end do
          n=0
      do while (Cp.ge.0.48+0.02*n)
                n=n+1
          end do
          x1=interp(c(n,m),c(n+1,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          x2=interp(c(n,m+1),c(n+1,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          c1R225=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
          x1=interp(c(n+12,m),c(n+13,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          x2=interp(c(n+12,m+1),c(n+13,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          c1R300=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
          x1=interp(c(n+24,m),c(n+25,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          x2=interp(c(n+24,m+1),c(n+25,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          c1R375=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
          open(15,file=file2,status='old')
          do 2 i=1,36
                read(15,*) (c(i,j),j=1,16)
2     continue
      close(15)
          x1=interp(c(n,m),c(n+1,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          x2=interp(c(n,m+1),c(n+1,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          c2R225=interp(x1,x2,0.5-(m-1)*0.1,0.5+m*0.1,R)
          x1=interp(c(n+12,m),c(n+13,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          x2=interp(c(n+12,m+1),c(n+13,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          c2R300=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
          x1=interp(c(n+24,m),c(n+25,m),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          x2=interp(c(n+24,m+1),c(n+25,m+1),0.48+0.02*(n-1),0.48+0.02*n,Cp)
          c2R375=interp(x1,x2,0.5+(m-1)*0.1,0.5+m*0.1,R)
          CR225=interp(c1R225,c2R225,k*0.001,(k+1)*0.001,Cv)
          CR300=interp(c1R300,c2R300,k*0.001,(k+1)*0.001,Cv)
          CR375=interp(c1R375,c2R375,k*0.001,(k+1)*0.001,Cv)
          FF=4./3*(Cbt-3)
          CRTSS=CR300+FF*(CR375-CR225)/2+FF**2*((CR375+CR225)/2-CR300)
          return
          end

          real function interp(y1,y2,x1,x2,x)
! This function performes linear interpolation.
          interp=y1+(y2-y1)*(x-x1)/(x2-x1)
          return
          end

          subroutine availspac(Cp,Crd,Vht,CN,Dav)
! This subroutine calculates the available space in the ship.
          real LWL
```

```fortran
        common/dimen/LWL,Vfl
        common/dimen1/T,B,HDK,D10
        common/raised/Hr
        Hr=9.5
        D0=2.011827*T-6.36215e-6*LWL**2+2.780649e-2*LWL
        D20=0.014*LWL*(2.125+1.25e-3*LWL)+T
        F0=D0-T
        F10=D10-T
        F20=D20-T
        Apro=LWL/0.98*(F0+4*F10+F20)/6
        Fav=Apro/LWL
        Dav=Fav+T
        CN=LWL*B*Dav/1e5
        Cw=0.278+0.836*Cp
        Vhaw=Fav**2*tand(10.0)*LWL+LWL*B*Cw*Fav
        Bmax=B+2*tand(10.0)*Fav
        Blow=Bmax-2*tand(10.0)*Hr
        Vhl=Crd*LWL*(Bmax+Blow)/2*Hr*Cw
        Vht=Vfl+Vhaw-Vhl
        write(10,*)
        write(10,1) 'Space Available ...'
        write(10,*)
        write(10,2) 'D0',D0,'[ft]'
        write(10,2) 'D10',D10,'[ft]'
        write(10,2) 'D20',D20,'[ft]'
        write(10,3) 'Total hull volume',Vht,'[ft 3]'
 1    format(2x,a)
 2    format(2x,a,t45,f5.2,x,a)
 3    format(2x,a,t45,f9.1,x,a)
      return
        end

        subroutine electric(Vht,VD,Pi,Wcps,KW24avg,Vaux,KWgreq)
! This subroutine calculates electrical load and auxiliary machinery rooms
! total volume.All loads in [KW].
      real LWL,KWp,KWs,KWe,KWm,KWcps,KWb,KWf,KWhn,KWa,KWserv,KWnp,KWpay
        real KWmfl,KWh,KWv,KWac,KWmflm,KWgreq,KW24,KW24avg
        common/dimen/LWL,Vfl
        common/dimen1/T,B,HDK,D10
        common/man/NO,NE,NA
        common/elect/Ng
        EDMF=1.0
        EFMF=1.01
        E24MF=1.2
        KWpay=947.8
        KWp=0.00323*Pi
        KWs=0.00826*LWL*T
        VT=Vht+VD
        KWe=0.000213*VT
        KWm=101.4
        KWcps=0.0
        if(Wcps.gt.0.0) KWcps=0.000135*VT
        NT=NO+NE
        KWb=0.235*NT
        KWf=0.000097*VT
        KWhn=0.000177*Vht
        KWa=0.65*NT
        KWserv=0.395*NT
        KWnp=KWp+KWs+KWe+KWm+KWb+KWf+KWhn+KWa+KWserv
! Iterative loop for net electrical load and AMR volume.
      Vmb=138620.0             ! MMR volume.
        KWmfl=3000.0           ! First guess.
 1      Vaux=56900.0*KWmfl/3411.0
        KWh=0.00064*(VT-Vmb-Vaux)
        KWv=0.103*(KWh+KWpay)+KWcps
        KWac=0.67*(0.1*NT+0.00067*(VT-Vmb-Vaux)+0.1*KWpay)
        f=KWnp+KWh+KWv+KWac+KWpay
        if(abs((KWmfl-f)/KWmfl).gt.0.01) then
             KWmfl=f
             goto 1
        endif
        KWmflm=EDMF*EFMF*KWmfl
        KWgreq=KWmflm/(Ng-1)/0.9
        KW24=0.5*(KWmfl-KWp-KWs)+KWp+KWs
        KW24avg=E24MF*KW24
```

```
         write(10,*)
         write(10,2) 'Electrical Loads ...'
         write(10,*)
         write(10,3) 'Winter cruise electrical load',KWmfl,'[KW]'
         write(10,3) 'Marginal winter cruise electrical load',KWmflm,'[KW]'
         write(10,3) '24 hours average electrical load',KW24avg,'[KW]'
         write(10,3) 'Power required per generator',KWgreq,'[KW]'
         write(10,4) 'Auxiliary machinery rooms volume',Vaux,'[ft^3]'
2     format(2x,a)
3     format(2x,a,t45,f6.1,x,a)
4     format(2x,a,t45,f7.1,x,a)
      return
         end

         subroutine tank(E,Ve,Vmax,SHPe,KW24avg,Pi,Vtk)
! This subroutine calculates fuel weight and volume of all tanks.
      common/liquid/WF46,WF52,WF41
         common/man/NO,NE,NA
         real KW24avg
         WF42=64.4
         NT=NO+NE
         Pebavg=1.1*SHPe/0.97
         rppe=Pebavg/(Pi/2.0)
      rnpe=Ve/Vmax
         rqpe=rppe/rnpe
         SFCpe=0.4097          ! [lb/hp*hr].
         SFCepe=SFCpe/rppe*(7.215e-2*exp(1.252*rnpe)+0.3629*rqpe*          &
           exp(0.7229*rnpe))
         f1=1.02
         if(1.1*SHPe.le.Pi/6) f1=1.04
         if(1.1*SHPe.ge.Pi/3) f1=1.03
         FRsp=f1*SFCepe
         FRavg=1.05*FRsp
         Wbp=E/Ve*Pebavg*FRavg/2240.0                   ! [ltons].
         Pg24=KW24avg/(0.7457*0.961*0.999)
         rpg=Pg24/9200.0
         rng=1.0
         rqg=rpg/rng
         SFCg=0.4727           ! [lb/hp*hr].
         SFCg24=SFCg/rpg*(0.2821+0.7179*rqg)
         SFCge24=SFCg24*Pg24/KW24avg
         FRgsp=1.04*SFCge24
         FRgavg=1.05*FRgsp
      Wbe=E/Ve*KW24avg*FRgavg/2240.0
         WF41=(Wbp+Wbe)/0.95
         Vf=1.02*1.05*42.3*WF41
         Vhf=1.02*1.05*43.0*WF42
         WF46=17.6
         Vlo=1.02*1.05*39.9*WF46
         WF52=NT*0.15
         Vw=1.02*36*WF52
         Vsew=(NT+NA)*2.005
         Vwaste=0.02*Vf
         Vbal=0.19*Vf
         Vtk=Vf+Vhf+Vlo+Vw+Vsew+Vwaste+Vbal
         write(10,*)
         write(10,1) 'Tankage ...'
         write(10,*)
         write(10,2) 'Fuel weight',WF41,'[lton]'
         write(10,3) 'Fuel tanks total volume',Vf,'[ft 3]'
         write(10,3) 'Ballast tank volume',Vbal,'[ft 3]'
         write(10,3) 'Total tanks volume',Vtk,'[ft 3]'
1     format(2x,a)
2     format(2x,a,t45,f6.1,x,a)
3     format(2x,a,t45,f7.1,x,a)
      return
         end

         subroutine weight(Wfl,Pi,Vt,LWL,CN,Ts,Ng,KWg,VD,Dav,KG,NHPIE,      &
           ERRw,Hmb)
! This subroutine calculates the weight (full load and light weight).
      real LWL,KWg,KGmarg,KG
         common/weig/Wops
         common/liquid/WF46,WF52,WF41
         common/prop/Dp
```

```
common/man/NO,NE,NA
common/dimen1/T,B,HDK,D10
WMF=0.005
KGmarg=0.5
Wvp=334.8
VCGvp=23.7
Wp100=81.2
VCGp100=40.1
Wp400=176.5
VCGp400=45.5
Wp500=15.7
VCGp500=32.2
Wp600=0.0
VCGp600=0.0
Wp700=314.6
VCGp700=35.3
W498=86.5
VCG498=-3.95
W237=0.0
VCG237=0.0
Wbm=P1*(9.0+12.4*(P1*1e-5-1)**2)/2240.0
Ws=0.41*LWL
Wpr=0.174*Dp**(5.497-0.0433*Dp)/2240.0
Wb=0.235*(Ws+Wpr)
Wst=Ws+Wb+Wpr
W2=Wbm+Wst+W237
W3=50.0+0.0352*Ng*KWg
W1c=4.45e-5*Vt
Wco=2.2*CN
Wcc=0.4*(Wp400+W1c+Wco)
W4=Wp400+W1c+Wco+Wcc+W498
W598=62e-6*Vt
NT=NO+NE
Waux=(0.000772*Vt**1.443+5.14*Vt+6.19*Vt**0.7224+377.*NT+2.74*P1)&
  *1e-4+117.0
W593=10.0
W5=Waux+Wp500+W593+W598+Wcps
Wofh=4.18e-4*Vt
Wofp=0.8*(NT-9.5)
W6=Wofh+Wofp+Wp600
W7=Wp700
Wbh=0.93*1.135*(1.68341*CN**2+167.1721*CN-103.283)
Wdh=0.00168*VD
W171=2.0
W180=0.0735*(Wbh+W2+W3+W4+W5+W6+W7)
W1=Wbh+Wdh+W171+W180+Wp100
Wm24=WMF*(W1+W2+W3+W4+W5+W6+W7)
W1s=W1+W2+W3+W4+W5+W6+W7+Wm24
WF31=NT*2.45e-3*Ts
WF32=0.00071*Ts*NT+0.0049*NT
WF10=(236*NE+400*(NO+1))/2240.0
Wt=W1s+Wvp+WF41+WF46+WF52+WF31+WF32+WF10
ERRw=abs((Wt-Wfl)/Wt)
if(ERRw.gt.0.01) then
      Wfl=Wt
      return
endif
VCGbh=0.51*D10
VCGdh=D10+1.5*HDK
VCG180=0.5*D10
VCG171=D10+0.13*LWL
P100=Wbh*VCGbh+Wdh*VCGdh+W180*VCG180+W171*VCG171+Wp100*VCGp100
VCGbm=0.5*D10
VCGst=4.8+0.35*T
P200=Wbm*VCGbm+Wst*VCGst+W237*VCG237
VCG300=0.55*D10
P300=W3*VCG300
VCG1c=D10
VCGco=5.6+0.4625*D10
VCGcc=0.5*D10
P400=W1c*VCG1c+Wco*VCGco+Wcc*VCGcc+W498*VCG498+Wp400*VCGp400
VCGaux=0.9*(D10-9.4)
P500=Waux*VCGaux+Wp500*VCGp500
VCGofh=0.65*D10
VCGofp=4.2+0.4*D10
```

```
          P600=Wofh*VCGofh+Wofp*VCGofp+Wp600*VCGp600
          P700=Wp700*VCGp700
          Pwg=P100+P200+P300+P400+P500+P600+P700
          VCGls=Pwg/(Wls-Wm24)
          VCGF10=0.732*D10
          VCGF31=0.523*Dav
          VCGF32=0.592*Dav
          VCGF41=10.3
          Hmb=D10-NHPIE*HDK/2
          VCGF46=0.53*Hmb
          VCGF52=0.138*Dav
          Pwgl=WF10*VCGF10+WF31*VCGF31+WF32*VCGF32+WF41*VCGF41+WF46*VCGF46+&
            WF52*VCGF52+Wvp*VCGvp
          KG=(Pwg+Pwgl)/Wt+KGmarg
      write(10,*)
          write(10,1) 'Weight and Center of gravity ...'
          write(10,*)
          write(10,2) 'Lightweight',Wls,'[lton]'
          write(10,3) 'Vertical CG of lightship',VCGls,'[ft]'
          write(10,2) 'Full load displacement',Wt,'[lton]'
          write(10,3) 'Vertical CG at full load',KG,'[ft]'
 1    format(2x,a)
 2    format(2x,a,t45,f7.1,x,a)
 3    format(2x,a,t45,f5.2,x,a)
      return
          end

          subroutine space(Npeng,Ng,Ts,CN,Vht,Vaux,Vtk,VD,Atr,Ata,Adr,Ada)
! This subroutine calculates space requirements.
      common/man/NO,NE,NA
          common/dimen1/T,B,HDK,D10
          common/duct/NDIE,NHPIE,NHeIE
          ADPA=578.0
          ADPC=3464.0
          AHPA=5173.0
          AHPC=6292.0
          AIE=159.1
          AGIE=35.7
          Adpr=1.15*ADPA+1.23*ADPC
          Ahpr=1.15*AHPA+1.23*AHPC
          Acoxo=225.0
          Ado=75.0*NO
          Adl=Acoxo+Ado
          Ahab=50.0
          NT=NO+NE
          Ahl=Ahab*(NT+NA)-Adl
          Ahs=300.0+0.0159*NT*9*Ts
          Adm=0.05*(ADPR-Adl)
          Adb=16*(B-19.0)
          Ahsf=1750.0*CN
          Apie=Npeng*AIE
          Aeie=Ng*AGIE
          Adie=1.4*NDIE*(Apie+Aeie)
          Ahie=1.4*(NHPIE*Apie+NHeIE*Aeie)
          Ahr=Ahpr+Ahl+Ahs+Ahsf+Ahie
          Vhr=HDK*Ahr
          Adr=Adpr+Adl+Adm+Adb-Adie
          Vdr=HDK*Adr
          Atr=Ahr+Adr
          Vtr=Vhr+Vdr
          Vmb=138620.0
          Vha=Vht-Vmb-Vaux-Vtk
          Aha=Vha/HDK
          Vta=Vha+VD
          Ada=VD/HDK
          Ata=Aha+Ada
          write(10,*)
          write(10,1) 'Area/volume Balance ...'
          write(10,*)
          write(10,2) 'Required hull area',Ahr,'[ft 2]'
          write(10,2) 'Available hull area',Aha,'[ft 2]'
          write(10,3) 'Required hull volume',Vhr,'[ft 3]'
          write(10,3) 'Available hull volume',Vha,'[ft 3]'
          write(10,2) 'Required deckhouse area',Adr,'[ft 2]'
          write(10,2) 'Available deckhouse area',Ada,'[ft 2]'
          write(10,3) 'Required deckhouse volume',Vdr,'[ft^3]'
          write(10,3) 'Available deckhouse volume',VD,'[ft^3]'
          write(10,2) 'Total required area',Atr,'[ft^2]'
          write(10,2) 'Total available area',Ata,'[ft^2]'
          write(10,3) 'Total required volume',Vtr,'[ft^3]'
          write(10,3) 'Total available volume',Vta,'[ft^3]'
 1      format(2x,a)
 2      format(2x,a,t45,f7.1,x,a)
 3      format(2x,a,t45,f8.1,x,a)
        return
          end

    subroutine stabil(KG,Cgmb)
! This subroutine calculates initial stability parameters.
        real KG,LWL,KB
          common/param/Cp,Cx,Cdl,Cbt,CD10,Crd
          common/dimen1/T,B,HDK,D10
          common/dimen/LWL,Vfl
          Cw=0.278+0.836*Cp
          Cit=-0.537+1.44*Cw
          KB=T/3*(2.4-Cp*Cx/Cw)
          BM=LWL*B**3*Cit/12/Vfl
          GM=KB+BM-KG
          Cgmb=GM/B
          write(10,*)
          write(10,1) 'Initial Stability .....'
          write(10,*)
          write(10,2) 'KB',KB,'[ft]'
          write(10,2) 'BM',BM,'[ft]'
          write(10,2) 'Metacentric height',GM,'[ft]'
          write(10,3) 'GM to B ratio',Cgmb
 1      format(2x,a)
 2      format(2x,a,t45,f5.2,x,a)
 3      format(2x,a,t45,f5.3)
        return
          end

          real function climbing(x,dx)
! This function guesses new value for the specified gene.
          integer r_dis(1)
          call rnund(1,2,r_dis)
          if(r_dis(1).eq.1) climbing=x+dx
          if(r_dis(1).eq.2) climbing=x-dx
          return
          end
```