OPTIMIZATION TECHNIQUES WITH KNOWLEDGE BASED CONTROL IN SHIP CONCEPT DESIGN

A Thesis submitted for the degree of Doctor of Philosophy

by

Richard David Schachter

Department of Mechanical Engineering, Brunel University

November 1990

To Maria Luiza Campinho P. Schachter, my wife

ABSTRACT

An integrated computational approach to Ship Concept Design using optimization techniques and a knowledge base to control the optimization process has been developed. The system automates both synthesis and analysis; analysis by the repeated sequential use of Design Theory Modules and synthesis through the optimization process, which compromises conflicting requirements, subject to constraints.

The intention of this work has been to find a better approach to automated design synthesis and at the same time employ detailed analytical tools such as a three-dimensional hull-form definition and engineering analysis modules.

Optimization techniques and a knowledge base are combined to achieve the desired capabilities, taking advantage of the benefits optimization can bring using goal oriented methods and exploratory searches, alongside a knowledge base that controls the synthesis process rather than the design. A function mapping strategy has been developed to provide a multiple-parametric view of regions of the optimization objective function and constraints. A discussion is included on the role of further applications of expert systems to design systems in both synthesis and analysis and their possible interference with creativity and innovation.

Two design examples are provided, one showing the application of the system using optimization and the other adding the use of the knowledge base. The results are compared and discussed.

CONTENTS

	page
Abstract	iii
Contents	iv
List of Figures	ix
List of Tables	xii
1. INTRODUCTION	1
1.1 Computer-aided ship design	1
1.2 Background - Historical Survey	4
1.2.1 Ship Design	4
1.2.2 Techniques for ship design synthesis	7
1.2.2.1 Optimization	7
1.2.2.2 Expert systems	9
1.3 Limitations of the techniques	10
1.4 The present investigation	13
1.5 Outline of the thesis	14
2. SHIP DESIGN	16
2.1 Introduction	16
2.2 Traditional Ship Design	18
2.1.2 Aims	18
2.2.2 Specifications	20
2.2.3 Concept and Preliminary Design (The Design Spiral)	21
2.3 The Evolution of Tools in Ship Design	24
2.4 Simulating Reasoning in Design - Requirements for a System	29

page

3. OPTIMIZERS 36

3.1 Introduction 36

3.2 Definitions 37

3.3 Optimization Techniques 40

3.4 Limitations 45

3.5 The Mapping Strategy 47

3.6 Summary 52

4. APPLICATION OF OPTIMIZATION TO 55
SHIP CONCEPT DESIGN - AN EXAMPLE

4.1 The Design System Structure 55

4.1.1 The Data-Base Handler (GENDAT) 57

4.1.2 The Design Control Module (CONSST) 58

4.1.3 The Optimizer (OPTCTL) 62

4.1.4 The Design Theory Modules 64

4.1.4.1 Three Dimensional Hull-Form Creation and Drawing (HULLCR-HULLDR) 65

4.1.4.2 Hydrostatic and Stability Calculations (STABCV) 69

4.1.4.3 Enclosed Volume Estimation (SPACE) 70

4.1.4.4 Lightship Weight, Condition Displacement and CG Estimates (WEIGHT) 70

4.1.4.5 Resistance Calculation (RESHLT) 73

4.1.4.6 Stability Criteria Verification (STCIMO) 75

4.2 Design Example 76

4.2.1 Selection of Objective Function, Constraints and Variables 78

4.2.2 Two Dimensional Mapping of the Objective Function 79

page

4.2.3 Optimization with Two Variables 80

4.2.4 Optimization with Five Variables 84

4.2.5 Analysis 88

5. KNOWLEDGE BASED SYSTEMS 101

5.1 Introduction 101

5.2 Definitions 101

5.2.1 Knowledge Based Systems 102

5.2.2 Artificial Intelligence 102

5.2.4 Expert Systems 104

5.2.3.1 Representation of Knowledge 107

5.2.3.2 Heuristic Search 108

5.2.3.3 Separating Knowledge from Inference and Control 109

5.3 Features that Characterise Expert Systems 110

5.3.1 A Simple Example 110

5.3.2 General Features 111

5.3.3 Knowledge Base and Inference Engine 112

5.4 Typical Areas of Application of Expert Systems 114

5.5 Expert Systems and Ship Design 117

5.5.1 Managing Specification (Input) Data Uncertainty 119

5.5.2 Example of Coupling Symbolic and Numerical Relations 121

- Interdependency of Parameters

5.5.3 Controlling Optimization 122

5.6 Limitations of the Technology 122

page

6. AN EXAMPLE APPLICATION OF A KNOWLEDGE BASE TO CONTROL OPTIMIZATION IN THE SHIP CONCEPT DESIGN SYSTEM 130

6.1 Ship Design optimization problems and control 130

6.2 The Leonardo System 131

6.3 Optimization Control using Leonardo - Knowledge Bases 1 and 2 134

6.3.1 Knowledge Base 1 134

6.3.2 Knowledge Base 2 136

6.3.2.1 Initialising - Mapping Strategy 137

6.3.2.2 The "Success" Branch 138

6.3.2.3 The "Failure" Branch 138

6.3.2.4 Using the 'Why' and 'How' Devices and Backchaining 141

6.4 Example Testing of Optimization Control using Knowledge Base 2 141

6.4.1 Modifications to the Constraints - Mappings 143

6.4.2 Modifications to the Initial Ship 145

6.5 Optimization example applying Knowledge Base 2 147

6.5.1 APPROX with two variables 147

6.5.1 APPROX with five variables - first run 148

6.5.3 APPROX at a closer starting point combined with step size tuning 150

6.5.4 Changing to SEEK 150

6.5.5 SEEK with OPTIM 1 151

6.5.6 SEEK with OPTIM 2 152

6.5.7 Random Search 153

page

6. AN EXAMPLE APPLICATION OF A KNOWLEDGE BASE TO CONTROL OPTIMIZATION IN THE SHIP CONCEPT DESIGN SYSTEM 130

6.1 Ship Design optimization problems and control 130

6.2 The Leonardo System 131

6.3 Optimization Control using Leonardo - Knowledge Bases 1 and 2 134

6.3.1 Knowledge Base 1 134

6.3.2 Knowledge Base 2 136

6.3.2.1 Initialising - Mapping Strategy 137

6.3.2.2 The "Success" Branch 138

6.3.2.3 The "Failure" Branch 138

6.3.2.4 Using the 'Why' and 'How' Devices and Backchaining 141

6.4 Example Testing of Optimization Control using Knowledge Base 2 141

6.4.1 Modifications to the Constraints - Mappings 143

6.4.2 Modifications to the Initial Ship 145

6.5 Optimization example applying Knowledge Base 2 147

6.5.1 APPROX with two variables 147

6.5.1 APPROX with five variables - first run 148

6.5.3 APPROX at a closer starting point combined with step size tuning 150

6.5.4 Changing to SEEK 150

6.5.5 SEEK with OPTIM 1 151

6.5.6 SEEK with OPTIM 2 152

6.5.7 Random Search 153

page

6.5.8 Optimization with five variables without using the Knowledge Base 154

6.6 Conclusions and Improvements 155

7. CONCLUSIONS 174

APPENDIX A OPTIMIZATION METHODS (OPTIVAR) 179

APPENDIX B Knowledge Base 2 (rule base only) 185

APPENDIX C 191

REFERENCES 199

ACKNOWLEDGEMENTS 207

List of Figures

		page
2.1	The Design Spiral.	35
4.1	Design system structure.	91
4.2	Body plan for the initial design.	91
4.3a	Variation of non-dimensional position of maximum beam, L'_X (aft).	93
4.3b	Variation of non-dimensional position of maximum beam, L'_X (forward).	93
4.4	Variation of waterline flare at max. beam, $FLARE_X$	93
4.5a	Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T.	94
4.5b	Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T (taken from Figure 4.5a, constrained).	94
4.6a	Contour map of total resistance, R_T (kN) at 15 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T.	95
4.6b	Detail contour map of total resistance, R_T (kN) at 15 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T (taken from Figure 4.6a, constrained).	95
4.7	Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L'_X	96
4.8	Contour map of total resistance, R_T (kN) at 30 kts. for waterline flare at maximum beam, $FLARE'_X$ versus waterline beam to draught ratio, B_{WL}/T	96
4.9a	Optimization path for APPROX with two variables.	98
4.9b	Optimization path for SEEK and OPTIM1 with two variables.	98
4.9c	Optimization path for SEEK and OPTIM2 with two variables.	99
4.10a	Body plan for the best, constrained design.	100
4.10b	Body plan for the initial design.	100
4.10c	Water-planes for the best, constrained design.	100

		page
4.10d	Water-planes for the initial design.	100
5.1a	Depth-first search.	129
5.1b	Breadth-first search.	129
5.2	Schematic view of an example-test program to relate parameters using an expert system shell.	129
6.1	Knowledge Base 1 (flow-chart).	159
6.2	Knowledge Base 2 (flow-chart).	160
6.3a	Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T for testship, unflared, unconstrained.	162
6.3b	Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T for testship, unflared, constrained.	162
6.4a	Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T for testship, flared, unconstrained.	163
6.4b	Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T for testship, flared, constrained.	163
6.5a	Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T contrained only by structural strength and the GM, unflared.	164
6.5b	Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T contrained only by structural strength and the stability curve areas, unflared.	164
6.5c	Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T contrained only by structural strength and the enclosed volume, unflared.	165
6.6a	Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T contrained only by structural strength and the GM, flared.	165

		page
6.6b	Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T contrained only by structural strength and the stability curve areas, flared.	166
6.6c	Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T contrained only by structural strength and the enclosed volume, flared.	166
6.7	Contour map of the metacentric height, GM (m) for waterline flare at maximum beam, $FLARE'_X$ versus waterline beam to draught ratio, B_{WL}/T, at ⓜ=7.0.	168
6.8	Contour map of the metacentric height, GM (m) for waterline flare at maximum beam, $FLARE'_X$ versus waterline beam to draught ratio, B_{WL}/T, at ⓜ=9.0.	168
6.9a	Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L'_X, at ⓜ=7.0, unflared.	169
6.9b	Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L'_X, at ⓜ=7.0, flared.	169
6.10a	Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L'_X, at ⓜ=9.0, unflared.	170
6.10b	Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L'_X, at ⓜ=9.0, flared.	170
6.11	Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T, at the "best found" region.	172
6.12	Contour map of total resistance, R_T (kN) at 30 kts. for waterline flare at maximum beam, $FLARE'_X$ versus waterline beam to draught ratio, B_{WL}/T, based on a ship with rounded up characteristics at the "best found" region.	172
6.13	Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L'_X, based on a ship with rounded up characteristics at the "best found" region.	173

List of Tables

		page
4.1	Initial input data.	92
4.2	Optimization Conditions (Summary).	92
4.3	Optimization with Two Variables.	97
4.4	Optimization with Five Variables.	97
4.5	Details of the Final Designs.	99
6.1	Initial input data.	161
6.2	Optimization Conditions (Summary).	161
6.3	Comparisons and details of the configurations with the final design.	167
6.4	Optimization using Knowledge Base 2.	171

1. INTRODUCTION

The classical book 'Basic Ship Theory', Rawson and Tupper[1] reminds us that the 'raison d'être' of the naval architect is the design of ships. Ship design, like all design, is regarded as a creative process where several, often competing, requirements must be met while a predetermined role is carried out. According to Calkins[2] design may be defined as "the arrangement of elements which obey laws so that an environment is created in which elemental interactions produce a desired result".

1.1 Computer-aided ship design

Ship design has traditionally been regarded as a craft, where innovative solutions occur as a consequence of using the existing knowledge in a creative way. Nevertheless, the requirements of modern life demand that designs be produced to greater levels of detail, in shorter time-scales and with fewer people than hitherto. The availability of powerful computing facilities has transformed the work of the average design engineer, including the naval architect.

Great improvements have been achieved in the development of computer systems, and this has stimulated further research and development into new computer based tools. The use of computer systems in design not only enables more work to be done in less time, but also allows different approaches to be adopted.

The early applications of computers focused on systemising repetitive tasks such as hydrostatic characteristics calculations, and then went on to disciplines involved in composing the ship design, such as hull fairing, strength and stability calculations, motions estimates, powering, manoeuvring, costing, etc. However, most of these programs are analytical rather than creative in nature, having been developed in isolation to deal with specific requirements of an essentially quantitative discipline. The application of computers to the synthesis of new designs has proven intractable. This is perhaps to be expected, given the difficulty of capturing the creative talent of the designer.

The identification of a distinction between synthesis and analysis has been important in helping to improve further the design process. This distinction has been identified by many authors, such as Andrews[3,4] and Calkins[2]; the former giving an extensive discussion in the context of preliminary design and the latter giving a comprehensive survey of systems and techniques recently adopted.

Clearly, the automation of design synthesis using a modular structure of analytical tools, that can be modified or substituted for, provides flexibility in the creative process and allows for more detailed calculations at the early stages of design.

Such detailed calculations are not always possible at the early stages of design due a to lack of relevant information at these stages. However, if somehow this could be overcome, it would be useful because it would help avoid many undesirable consequences that often occur,

after contract, in the detailed phase of design. Many contributions have been made describing computer-aided design systems that address these problems. Most of them integrate a number of the available analytical tools around data-bases; the analysis applied being concerned primarily with the use of mathematical models to produce further information (both geometric and numerical) to help extend the contents of the data-base. In such systems the expert performs the design, making the key decisions, leaving the computer to carry out supporting analytical calculations or to display and present the results.

Some developments have addressed systems that help, or even replace the expert by making decisions and/or by searching for successful combinations of ideas. A division can be made in such systems by distinguishing between the application of professional experience and the introduction of innovative ideas, arguing that the decision-making process requires reasoning and the use of knowledge, while the search for successful combinations of ideas draws on creativity. These two branches of the design process are respectively addressed by expert systems and optimization techniques, the former encompassing methods for storing knowledge while the latter tackles the problems of comparative testing of ideas.

Recently, most interest seems to have been concentrated on expert systems and a large amount of work is being carried out in this area. Expert systems are techniques that attempt to solve problems through symbolic, non-numeric representation, by explicitly representing both knowledge that an expert has about some domain and the strategies that

he uses to reason about this knowledge. This is achieved mainly through the use of knowledge bases, which separate rules from inference. Using this device, reasoning can be simulated and justified and expert knowledge can be captured, stored and used by less skilled operators.

Conversely, optimization is a process whereby various competing ideas are examined, using exploratory searches subject to constraints, by testing variations in some measure of merit. Here the aim is to make certain parameters as large or as small as possible although conflicts often arise, resulting in compromise situations. Optimization is by no means novel and although there seems to be less recent interest in applying it to ship design, new applications are emerging, making use of the benefits that the latest evolutions in computing are bringing.

1.2 Background - Historical Survey

1.2.1 Ship Design

The evolution of ship design as a scientific approach dates back to 1746[1], when the foundations of many aspects of naval architecture where laid down by Pierre Bouger, 'Traité du Navire'. From then on much work has been done with great contributions mainly in specific disciplines such as resistance - William Froude (1868), ship wave generation, in work done by Lord Kelvin at about the same time, followed by that of Havelock (1908) and Hogner (1922).

Much more recently, several publications have become modern classics, some as a systematic view of ship design as a whole and others as contributions to specific areas, which have become widely used methods for ship designers. As has already been mentioned, there is Basic Ship Theory, by Rawson and Tupper[1] (1968) and also Principles of Naval Architecture, edited by Comstock[5] (1967) as well as papers such as Lamb's[6] (1969) and Watson and Guilfillan's[7] (1976), the latter being a review of their design methods following rapid changes due to the development of computer technology. Certain methods have become widely accepted and used due to their simplicity, and ease of understanding by engineer designers, while providing enough accuracy for the needs of preliminary design. Some of these are statistical methods for resistance evaluation, as surveyed by Oossanen[8], of which Holtrop's[9,10,13] and Holtrop and Mannen's[11,12] and Oortmerssen's[14] are most commonly used, the first for ships in general and the second for small vessels; or in the case of stability, Sarchin and Goldberg's[15] work for warships that has proved to be so universal and comprehensive that it can be applied to any type of floating craft. Such simplified, but efficient theories have helped give insight into systemising the design process.

According to Farrar[16], ship design with the help of computers dates back at least 25 years to early work by the British Ship Research Association (BSRA) using mainframe computers which could only communicate in numbers by keyboard and printer, with no visual display. In the late 1970's developments in microchip technology brought the mini and then the micro-computer, which allowed for a dramatic evolution in programming, due to the much reduced costs of the computing hardware which made it

economically possible to spend much more on the development of the corresponding software, since many more users could afford a computer than just a number of skilled programmers.

A large number of publications has consolidated the use of computers in ship design, some representing important steps in this process. In 1972, Snaith and Parker[17] looked at the scope for the use of computers in ship design, concluding that only then was the potential use of computers becoming fully appreciated. In their paper 'Concept Exploration - an Application to Small Warship Design', Eames and Drummond[18], following the tendency of the time, introduced the idea of an integrated computer system where they criticised the purely analytical approach of the time and looked for ways of finding an appropriate design which was not necessarily based on some existing one. This system was still under manual control and did not deal with three-dimensional hull-forms. An integrated computer system whereby a mini-computer that could be used interactively during the early stages of warship designs was presented by Yuille[19] in 1978. This system allowed for three-dimensional displays and modifications of the hull-form and equipment arrangements on the disk store as the design developed with analytical calculations. Parsons and Beier[20] (1987) developed a computer-aided ship design system for use on micro-computers for academic purposes which deals with a three-dimensional hull-form and which has graphical abilities. In 1983, Calkins[21] presented a computer-aided design system for recreational powerboats based on the design spiral concept and made an attempt to synthesise the design process, using design analysis modules, giving graphical output and allowing for interactive use with the designer.

Gradually design systems have evolved for dealing with the synthesis process and the automation of this process seems to be the next goal to be achieved. Methods that generate a three-dimensional hull-form from a set of parameters, such as Keane's[22] allow for a more detailed approach to such automation of the design synthesis.

1.2.2 Techniques for ship design synthesis

1.2.2.1 Optimization

Optimization techniques have been used for a long time. Danzig first solved the linear form of optimization equations in 1947[23]. Nevertheless there are very few linear applications in engineering. A great deal of effort has gone into developing techniques for the non-linear case, but none have been as successful as the linear one, in the sense that a global optimum can be guaranteed in a finite number of steps. Despite this, there are a number of good methods that work successfully in most applications, such as Avriel, 1973 and 1976; Beveridge and Schechter, 1970; Fletcher, 1970; Fox, 1970; Gill and Murray, 1974[24,25]; Gottfried and Weisman, 1973; Kowalik and Osborne, 1968; Murray, 1972 and Siddall, 1972. These are summarised by Siddall[23].

Some new, more unconventional techniques have been developed recently, namely, the genetic algorithm technique, which simulates a Darwinian survival-of-the-fittest with randomised yet structured information exchange among a population of artificial chromosomes - Goldberg[26], Poole and Adams[27], Poole[28], 1988. In 1983, Kirkpatrick et al.[29]

developed a method that attempts to optimize by means of a simulated annealing process.

Among the applications of optimization to ship design, perhaps one of the most comprehensive works is that of Parsons[30] in 1975, where most of the classical methods that are applicable to ship design are introduced; a computer program for a few is presented and some examples are shown. The work also presents a survey of optimization applications to ship design, from 1965 to 1974.

There have been applications of optimization also to specific areas of naval architecture, such as Bales[31] for optimizing seakeeping performance of destroyers, but most of them were aimed at the ship design as a whole. Mandel and Leopold[32] developed in 1966 a method to be applied in preliminary ship design in a convergent random search technique using a weighted multiple-parameter optimization criterion that was intended to save time and to be more versatile than other methods; Beier et al.[33] (1976) derived a general purpose system for optimization in ship design where the optimizer dealt with the input and the output of design application programs, but the studies were for parametric calculations; and more recently, in 1985, Pantazopoulos[34] proposed an automated design optimization process combined with computer graphics for conceptual ship design, taking advantage of the new capabilities of computers to run faster, handle bigger programs and produce graphics. Lyon and Mistree[35] (1985) proposed a computer based method for preliminary design using a design optimization model that involves a mix of linear and non-linear goals and constraints in a

multiple design objective fashion, i.e. all design tasks are completed simultaneously, in one execution, and they make an interesting point that the process need no longer be interactive.

1.2.2.2 Expert systems

Expert systems are becoming increasingly popular in many sorts of applications, when symbolic, non-numerical problems are to be tackled. This technique causes great interest to designers because among other things, decision-making processes can be simulated. Some useful applications can be found in diagnosis problems and systems such as MYCIN, Buchanan and Shortliffe[36], among several others, have been developed and their use has given feedback to evolve the technique further. Waterman[37] gives a very comprehensive guide for these applications, while Johnson and Keravnou[38] examine their architecture; Hayes-Roth et al.[39] provide a good introduction to expert systems and insight on how to build them. Jackson[40] and Alty and Coombs[41] give a deeper understanding of their principles, while Harman et al.[42] put a practical view into context. In a compilation of papers, Kowalik[43] introduces work done in trying to solve problems by coupling symbolic and numerical computing. As a design example, Tong[44] also managed to couple, in his expert design system for aerodynamic bodies, a rule based system with computational fluid dynamic programs, to design an axial flow cooling fan.

Some progress has been made in applying expert systems to ship design. MacCallum[45] (1982) and MacCallum and Duffy[46] (1985) have developed the

DESIGNER system for preliminary design modelling using an expert system which relates ship design parameters and in which the knowledge for each characteristic is encapsulated in a frame containing relevant information on the parameters and how they relate to others (including the 'strength' of their influences). Welsh et al.[47] at the University of Newcastle have developed an expert system specific to container ship design in collaboration with British Shipbuilders where the steps of the ship design process are controlled by a knowledge base. This work was a further development of earlier studies carried out by the same group, for integrating ship design and production considerations; see W. Hill et al.[48] (1989).

1.3 Limitations of the techniques

It is quite rightly argued that there is no reliable and foolproof design synthesis method available. The use of optimization techniques, long abandoned by some, seems to be a very mechanistic way of synthesising a complex process such as design, relying as it does, on essentially numerate analysis. Expert systems look more attractive when the application is meant to simulate human intelligence, such as in the case of decision-making. However, it should be noted that innovative solutions are more likely using optimization methods (i.e., they test previously untried combinations of parameters), especially when compared to expert systems approaches which tend, as they are meant to, to produce designs that are very similar to those of an experienced engineer (i.e., they use the expert's rules-of-thumb and are evolutionary in nature). Another particular strength of optimization

methods lies in their ability to perform goal oriented tasks; targets can be specified and optimizers deployed so as to drive a design towards the desired goals. This is much more difficult to institute with a rule based regime, primarily because such goals are essentially specified numerically and in order to make such improvements, the requirement of calculating variations in some measure of merit arises. When using optimization, systematic variations of parameters are tried, in order to achieve a certain goal which would minimise or maximise parameters, subject to constraints. Of course, once a mechanism for improvement has been found it can be pursued until exhausted. This should be contrasted with knowledge based methods which try to direct the design process by reference to previously successful design rules.

The exploratory search of an optimizer is, of course, a much longer process, because it tries possibilities regardless of whether human reasoning would recognise them as absurd. Usually, knowledge based structures lead more directly to the best design; also they are capable of indicating why various design decisions have been taken. The only justification open to optimizers, on the other hand, is that the final design meets the specified requirements better than all the other possibilities tested. However, if the design is to encompass the most reliable analytical tools, the resulting design problem becomes one of great complexity, involving many subtle interdependencies. Under such circumstances a knowledge based structure begins to make simplifications and to require the insertion of more rules, i.e., more knowledge. Moreover such rules are difficult to make universal as problems grow in complexity, requiring new rules for each new design problem.

Optimization techniques are time consuming, hard to understand and, if used in a traditional manner, the user is often uncertain of the results. In summary, no one approach is likely to provide all the answers, each having its own peculiarities:

(1) Traditional CAD systems are very flexible leaving the designer complete freedom in the decision-making process but requiring great skill to use well; the synthesis must be done by an experienced expert.

(2) Optimization methods are capable of extracting the most accurate information from the available design theory, but applying it to meet specific goals can be very time consuming, difficult to understand, and there is uncertainty as to whether the results are the best possible ones.

(3) Expert systems neatly distil the available knowledge and apply it in a fully explained fashion, but if the design theory is part of the knowledge base the system will be narrow in scope and detailed in knowledge. Such a structure will be able to give reasoning about the knowledge used but it may need to simplify the knowledge itself. It will also constrain the design, rather than the techniques that control it, making it easy to correct some decisions, but hard to change the theory.

The best approach ought to encompass a combination of these ideas, i.e., the aim ought to be to structure a CAD system in such a way that

detailed calculations can be run using exploratory searches, with guidance provided by an expert system using simple rules, the whole system being managed by the user via a powerful and flexible interface. This would allow the designer to vary both the measure of merit and/or the parameters used to achieve a good design as the process advances, or even to take manual control.

1.4 The present investigation

The intention of the present investigation is to provide a further step in computerising the design process, looking for a better approach to automated design synthesis at the same time that detailed analytical tools such as a three-dimensional hull-form definition and engineering analysis modules are available for use and manipulation without interfering with the synthesis structure. To aid in this study an integrated computational approach to ship concept design using optimization techniques with a knowledge base to control the optimization process has been developed.

The attempt here is to combine techniques such as optimization and knowledge based systems to achieve the desired capabilities, taking advantage of the benefits optimization can bring with goal oriented methods and exploratory searches, now made viable by the state-of-art of computer technology, including graphical facilities and using a knowledge base to control the synthesis process rather than the design. The knowledge base gives guidance on how to proceed with the design and to help overcome the classical problems that have made optimization

unattractive until the recent past: the uncertainty felt by the user due to lack of visibility of the function and the time consumed in vast exploratory searches. This is achieved using a mapping strategy that provides a multiple-parametric view of regions of the objective function and constraints.

Two design examples are provided, one showing the application of the system using optimization and the other adding the use of the knowledge base. The results are compared and discussed.

1.5 Outline of the thesis

In Chapter 2 the aims and the requirements of traditional ship design are discussed as well as the place of the Design Spiral in concept and preliminary design. Then the major tools of the subject are addressed in the context of historical and present day problems and needs. Requirements for a system that would simulate reasoning in the design process, using detailed theory at early stages, are proposed.

In Chapter 3 optimizers are presented. Some definitions are provided and some optimization techniques are introduced. The limitations of optimization and a discussion of how to overcome them are addressed and a mapping strategy for the design function is established.

In Chapter 4 a ship concept design system which automates both synthesis and analysis using function mapping strategies and optimization techniques is described. The system permits full design

integration using a three-dimensional hull-form definition as well as various design theory modules. In the second half of the Chapter a design example is presented. From the discussion of the results, the need for an expert system to control optimization and the mapping strategy is established.

In Chapter 5 Artificial Intelligence and Expert Systems are introduced and several definitions are provided. Areas of application are identified, among them some for design. A few example applications for feasibility verifications are described. At the end of the Chapter the role of the technique in the context of design is discussed along with its advantages and limitations and their possible interference with creativity and innovation.

In Chapter 6 a knowledge base designed to control optimization is described. The second half of the Chapter is dedicated to a further design example. From these results, both designs are compared and the advantages, limitations and further developments required of the applications are discussed.

In Chapter 7 a summary of the work is provided and the final conclusions, contributions and further developments addressed.

2. SHIP DESIGN

2.1 Introduction

Perhaps the best way to describe ship design is to use Rawson and Tupper's[1] definitions: "Ships are designed to meet the requirements of owners or of war and their features are dictated by these requirements. The purpose of a merchant ship has been described as conveying passengers or cargo from one port to another in the most efficient manner". They go on to remind us that the economics of any particular market have a profound effect on merchant ship design. Economic reasons are decisive factors for whether more or less cargo is needed per trip, the choice of operational speed or the best configurations for various ranges. Handling, harbour, governmental, etc. aspects can also play a key role in the sizing of a ship. The type of cargo they are designed to handle obviously makes the ships differ substantially.

Different missions require different ships. Mission definitions can take various criteria into account besides economics. Warships, for instance, are defined by governments' defence policies and vary substantially according to their missions. In war the invention of a certain type of warship leads to the design of another to neutralise its offensive abilities. For a tug or a trawler, the bollard pull and manoeuvrability are dominant features, while with a hydrofoil boat the main concerns are with the operating seastate and hydrodynamic lift. A

hovercraft design would require aerostatic lift calculations and would have to focus strongly on course keeping abilities. The geometry and propulsion of an ocean going ship would be completely different from those of a river ship operating in shallow waters. In SWATHS, if one is to benefit from their superior stability, seakeeping, etc., careful studies, with attention to structural weight, engine room arrangements and manoeuvrability considerations must be made.

It is obvious that general arrangements and compartmentation of different types of ships follow completely different criteria. The space allocation and concept of comfort in terms of space and heave accelerations of a warship are different from those of a passenger or a hospital ship. The compartmentation of a container ship or a tanker, which differ significantly, must maximise space allocation following a different criteria from that of a frigate, which must be designed to carry and operate weapons and provide a strategic compartmentation with floodable sectors for passive protection against enemies' weapons. A supply boat has to carry and segregate industrial water and mud, drinking water, oil, cement, etc., all cargoes with different densities at extreme volume variations, with very little room for ballast to control the trim.

2.2 Traditional Ship Design

2.2.1 Aims

So what is good general design practice? As has been noted, the features that the naval architect has to take into account and compromise vary very much, but they can be generalised and systematised in design processes that look at the overall features as produced by the various parameters making up the design.

According to Rawson and Tupper[1] the naval architect is concerned with ship safety, performance and geometry although, as we can see, these are not exclusive divisions.

With ship safety, the concerns are that the ship does not capsize in a sea-way, or when damaged or even when maltreated. To ensure this stability calculations are carried out and checked by criteria that vary according to the type of ship and/or mission and operations involved. Seakeeping studies also provide data and information for stability, structural and survivability safety. Another fundamental requirement for safety is that the ship must be sufficiently strong so that it does not break up or fracture locally and let water in. This is ensured by carrying out structural and vibrational analysis, and/or following rules of good practice laid down by classification societies. Besides this the crew should be given a good chance of survival if the ship does let water in through accident or enemy action. This is affected by safety equipment, life rafts, fire fighting, etc.

The performance of a ship is dictated by the needs of trade or war. The required amount of cargo must be carried to the places which the owner specifies in the right condition and in the most economical manner; the warship must carry the maximum hitting power of the right sort and an efficient crew to the remote parts of the world. Size, tonnage, deadweight, endurance, speed, life, resistance, methods of propulsion, manoeuvrability seaworthiness, systems operations and other features must be matched to provide the right primary performance at the right cost.

Ship geometry concerns the correct interrelation of compartments which the architect of a house considers on a different scale. Each type of vessel, according to its operations, requires various different rooms or compartments to be related, in the most efficient way. Various systems (equipment, piping and ducting, etc.) must be correctly installed to allow their proper functioning. The architecture of the ship must be such that it can be economically built, and production arrangements for the ship are an important consideration. The builders' capabilities and limitations must be taken into account to enable the planned balance of features to be correctly constructed. Most important of all, the hull-form must balance all these features with hydrodynamic and hydrostatic performance and safety requirements such as resistance, propulsion, manoeuvrability, seakeeping, stability, freeboard that will also influence the size and will "sculpt" the external shape. Finally, the geometry must be arranged, in so far as possible, to be aesthetically pleasing, to appeal to its customers if it is a merchant ship or to intimidate potential enemies, if it is a warship.

In summary, what the naval architect needs is to create a geometry attached to a general configuration to meet a certain performance, satisfying safety standards at determined costs. He must satisfy design requirements while many aspects of the final configuration must be guaranteed at stages where there is little information available.

2.2.2 Specifications

Ships are always designed to a set of requirements or specifications. Every type of vessel has its own specialised features for the mission it is required to perform. The requirements may come from the shipowner who provides a comprehensive specification[17], detailed at various different levels, or just a specific mission definition; or they may be part of a fully planned complex operation where the role is determined as a consequence of a complete transport or operation system study. It can also be the case that, using marketing studies, the designer can identify potential customers for a certain type of vessel which he then tries to standardise in a concept, allowing for flexible changes of specification during contractual negotiations.

Besides the clients' expectations and specifications, production capabilities must also be matched to allow for the construction of the ship. Factors[17] such as available resources, physical setting, social context, available materials and the present state of technology to process them, economic factors, time scales, etc. must be considered.

Ship design also has to take into consideration safety, environmental and legal requirements (and sometimes even political ones) not only to satisfy the owner's desire for profitability and/or performance and the designer's understanding of naval architecture, but also for the sake of approval by governmental bodies and classification societies in order to meet insurance standards. Governmental bodies, classification societies and international bodies lay down standards of good design and construction practice covering aspects such as rules for structural calculations and construction, tonnage regulations for taxation, stability criteria, measures for safety at sea, freeboard regulations, regulations for cargo handling, pollution at sea, etc. These are all major constraints that must be taken into account before concept studies are made. Such investigations will be called feasibility studies here.

2.2.3 Concept and Preliminary Design (The Design Spiral)

How can all these features be balanced to create such a complex product? Traditionally[17] most ship designs are based on previous experience using similar designs with modifications introduced to suit each new owner's particular requirements. Usually the design process starts with parametric surveys and studies based on existing ships to choose the main particulars needed to fulfil the requirements. There are, of course, several disciplines in common with any type of ship or mission for which calculation methods and drafting procedures should be readily available. These are the analytical tools which will help to build up and validate that the concept will achieve the required performance, subject to the relevant constraints.

Every preliminary ship design needs a general arrangement and lines plan, from which the hydrostatic characteristics can be found for all types of further calculations. The process can be highly interactive, and different levels of detail will be considered sufficient for each phase of the design process. To be able to put up a tender, the designer must get several features right. By way of example, it would be disastrous if, say, in a bulk carrier preliminary design there was an ill defined hull-form and power prediction that would eventually cause the need of an extra cylinder for the main engine, with this only being found out after the towing tank tests were made. Or the discovery that the vertical centre of gravity of a RO/RO passenger ship is excessively high during the light weight calculations of the detailed design, causing a whole redefinition of the general arrangement and the capacity plan.

In ship design cost must also, of course, be taken into consideration. It is very difficult to determine at once the right balance of requirements and the designer must go into more details iteratively as the needs arise. One way to control this process, that has already become a classical illustration of the design concept, is the design spiral[17,49] (Fig. 2.1, after Buxton in 1972). As mentioned by Snaith[17], the spiral concept of ship design may be extended to show that the design can be developed at several levels of complexity, depending on the purpose. A large number of designs may be examined on the early cycles, but usually only one on the final cycle. Each point of the spiral represent specialised calculations or considerations and these are essentially analytical procedures. The integration of the whole,

after each cycle, represents the synthesis in the process. Several iterations are needed to adjust the results, and these usually increase in complexity. During concept design the ship is defined as a whole while after preliminary design it should be ready for tender, with costs known and enough data for starting the detailed design for construction, after which only minor adjustments can take place.

There are several different definitions for stages of the traditional ship design process, focusing on different levels of detail. One classification of the ship design synthesis process, broken into stages, is given by Calkins[2] and this comprises of five stages, which are considered by the U.S. Navy as:

(1) Feasibility Studies - conducted to establish cost/performance trends i.e., cost of speed, endurance, payload, etc.

(2) Conceptual Design - considers trade-offs at the ship design level.

(3) Preliminary Design - focuses on the subsystem level.

(4) Contract Design - deals with definition of the ship for the builder.

(5) Detail Design - working drawings.

As mentioned before, with the exception of stage (5) these stages are achieved by progressing in an orderly fashion through the various points of the design spiral, in many loops and not necessarily one per stage.

2.3 The Evolution of Tools in Ship Design

In the past naval architecture used to be a very tedious activity in which several devices, long since abandoned, had to be used to allow for dealing with ship geometry. Nevertheless it was, and still is, regarded as a very creative and artistic activity, which explains some of the motivation of naval architects. Both design and construction were regarded as crafts. Obviously construction came first as the concept of design was not clearly defined in people's minds in the past. Here we are examining the design aspects of the profession.

The design synthesis process is strongly expressed in the definition and drawing of the general arrangement and the lines plan of the ship being created. All the requirements are not, of course, solely defined by these drawings, but they are the strongest indications of them.

In ship design, trade-offs can be expressed geometrically and this is a strong artistic motivation in the creative sense. The naval architect will try to work out the best general arrangement as an architect would do, considering interrelations of compartments, operationability of systems, etc. While drawing the lines plan, he will have in mind all the features he must comply with, trying to get the slenderest ship possible, with a "good" streamlined flow, suitable for the design speed, will be worried about added mass or damping of the hull in a seastate or will be trying to compromise draught with propeller clearance, thinking of the flow over the rudders, keeping a good wake without wasting volume. He will try to keep weights and fluids as low

as possible and will take into consideration the possible longitudinal positioning of fluids while defining shapes if it is a small boat. An experienced naval architect will compromise slenderness with structural or even welding capabilities and limitations, and some times even with plate bending. Soon after, the naval architect has to convert the lines plan into the now classical representations that would allow further calculations: hydrostatic curves, Bonjean curves, cross curves of stability, etc. To do this in the past it was necessary to perform geometrical integrations of areas measured by planimeters in very tedious repetitive calculations. Since these latter activities did not require creativity they were the first to be systematised. For quite a long time these integrations and characteristics calculations have been carried out by computers.

As the use of computers had developed more and more different aspects of naval architecture have been coded for calculations. The evolution of state of the art research to provide useful and practical analytical tools has of course contributed immensely to the development of new calculation methods. As an example, not so long ago the reliable evaluation of resistance was such a problem that designers would develop their body plans based on systematic series (e.g., Series 60, Taylor, Cetena, etc.), which were groups of scale model forms previously tested in towing tanks. Current calculation methods allow for much greater attention to detail during the earliest stages of design.

Many achievements have been made in calculation methods which, in due course, have been introduced into the computerised world. They are still separate analytical tools though, i.e., they perform the analysis but the synthesis remains up to the designer or is the result of interactions between sections of the design team (i.e., naval architecture, structures, machinery and equipment groups).

At the same time that research has evolved reliable practical methods and computers have become more popular, faster and with greater memories and easier to use, ship construction, as a result of the world's economic crisis, started to decline, and, as a consequence, contracts started being more scarce. Mistree[35] cites the major consequences of this: as compared to the recent past more preliminary designs are not followed through to build. Therefore, no company can afford a large expenditure of time and money on preliminary design work. Single vessel classes have become common, as opposed to the past when designs could be "optimized" by small improvements from ship to ship and class to class[35]. There are now few similar vessels from which to extrapolate new designs; therefore, it is more difficult to get a large amount of accurate data on similar ships. Design staffs are reduced in size due to periods of lack of work[35] and when a good period returns there is no time to train or get good cohesion in design teams. Moreover, a simply adequate design may no longer be competitive in the open market. These factors, added to those concerning naval architecture already discussed, require preliminary designs to be carried out with greater levels of detail, on shorter time-scales and with fewer people.

Mistree goes on to say that a rational, computer-assisted method employing optimization techniques can overcome many of these problems since it is capable of finding an optimal solution much faster. Thus a considerable portion of the preliminary design work can be completed by a single, relatively inexperienced designer in a short period of time. Once the decision to proceed with the new vessel has been made the same system should then be used for investigating alternative concepts and for detailed analysis. These ideas, which have been put forward by several authors, essentially try to work out rational structures of computer programs using techniques such as optimization and/or knowledge based systems to perform both/either the exploratory searches and/or the knowledge oriented tasks mentioned in Chapter 1 (and of which applications will be discussed in Chapter 4 and 6, respectively).

In a general sense, useful definitions are given by Calkins[2] for both Computer-Aided Design and Computer-Aided Engineering:

Computer Aided Design - the process of geometric modelling includes the conception and synthesis of a system, such as a ship, using interactive graphics techniques to display and view the design. Three-dimensional wire-frame models are the typical display format, with fully shaded colour raster models a developing alternative. The designer describes the shape of a structure with a geometrical model constructed graphically on the screen. The representation is based on a mathematical model which is stored in the computer data-base for later

use (see Chapter 4). The model may be used for other CAD functions, or it may be recalled and refined by the engineer at any point in the design process.

Computer Aided Engineering - the engineering analysis of the design concept or geometric model, created using CAD. With keyboard commands the user may have the computer calculate, for example, weight, volume, stability curves and analysis, etc.

Various authors have developed systems that would both perform analysis and synthesis of the design, as has already been mentioned in Chapter 1. One of the tendencies on ship design evolution seems to be in the direction of trying to take advantage of traditional CAD and CAE systems, but automating them to gain the benefits of exploratory searches that optimization can provide. The creativity in decision making obtained using CAD or making drawings and having these designs tested with engineering analysis should be grasped and automated, provided this is tailored for flexible control. Knowledge based structures can be applied at various levels, but maybe the most useful ones are those that are general enough not to force the knowledge to be simplified and not to inhibit the designer's creativity.

2.4 Simulating Reasoning in Design - Requirements for a System

An ideal design system might be one that would try to simulate the way a naval architect reasons and controls the design process as it is normally carried out in a design office. Given a specification or mission definition of some sort, the designer is supposed to know what different pieces of theory to use and to have access to data on similar ships, to establish reasonable boundaries for parameters so as to start a search for the required geometry. He knows he must satisfy rules, regulations and requirements for safety, environment and working practice. All these features are reflected in calculations and drawings. He has to achieve a balance of the results of these calculations and the arrangements produced, which are often conflicting in nature, reaching a final configuration that will satisfy the requirements.

A useful design system should not only give expert advice on what to do, but should also perform the calculations using the different pieces of theory selected. Systems that only give advice tend to lack detailed knowledge in this area, which arises as a consequence of using and analysing detailed calculations and then trying to compromise the results with other conflicting ones. A system that would cope with such theory would clearly be quite useful.

The idea of using detailed calculations in early stages of the design process has, perhaps, not been often used because at these early stages there is not enough information available to allow them to be performed.

However, if somehow this were possible and not too time or man hour consuming, no one could deny that it would be desirable to perform such calculations. The best way to make this possible is to use a database system in such a way that, if there is not enough information for a certain calculation, it will be replaced using default values or by asking for a designer input and then to control the consequences of this action later on.

The motivation for taking this strategic action is as follows:

(1) Computers now allow very fast processing, even for detailed calculations.

(2) Simpler formulations have been developed for the various disciplines that allow enough precision for concept and preliminary designs, such as statistical methods or simplified theoretical calculations, diminishing the need for heavy number crunching.

(3) Since the computing process can be systematic, it should require no extra time expenditure in man hours and can, from a certain stage, be run in a batch mode.

(4) Since the design process is interactive, the use of certain theory, if properly monitored along the design process:

(a) will tend to make the parameters involved converge;

(b) give directions, i.e., which parameters should be modified and how;

(c) set boundaries, i.e., eliminate useless areas of the design function;

(d) is a reminder of its own existence - even by giving erroneous results, at first, but giving, later on, the designer the comfort of knowing that nothing has been forgotten;

(e) may help explain far reaching consequences of variations in that context, which may not be easily seen and,

(f) even might tend to "give ideas" of improvements in the context of a complex combination of features, since the first thoughts of a designer tend to be for isolated phenomena and tendencies in separate areas of design.

The data-base structure forms a central store of information and separates the design processes, which are not design specific, from the design data, which relate only to the current task. This structure also helps in the testing of new ideas: if the designer thinks of something he needs to confirm this by calculation to see if it is worthwhile. He needs to see quickly what consequences for the whole process arise when slight modifications are made to one or a few parameters. Furthermore, most good designs are compromises and therefore tend to lie at the borders of critical constraints. Clearly if every feature, or as many as possible, are under control, one can safely afford to be less conservative, gaining a better overall performance and lower costs. However, stretching features to their limits while maintaining design compromises requires great expertise. Only experienced naval architects can visualise quickly in which directions they can vary parameters and by how much. Obviously there are advantages in being able to insert as many design features as possible

during the early stages. A system able to do this would tend to ensure the balance of conflicting requirements, and would not demand from the designer the great insight of what is fundamental for each particular case: in such a structure whatever is not important will simply not interfere.

Finally, the use of such a system would encourage the insertion of restrictions that normally are only in the back of the designer's mind. They might be transformed into a numerical representation or even a heuristic rule for a knowledge base and then be used as constraints or a control device, respectively.

Another important feature in the construction of design systems is sequencing: design theory modules should be able to run in a sequence that is as natural as possible, similar to that in the normal design spiral process. But since computer programs do not function as a spiral a consistency check must be carried out before each loop, so that the input values and/or defaults can go through the modules without causing inconsistencies to arise. Such checks allow a system to run many cycles on its own. Moreover, it is desirable for a system to make use of design theory modules that can cope with wide ranging parameters. A Design Control Module that non-dimensionalises geometrical parameters and that recycles the basic ones to get them consistent with the changes made solves this problem.

When running such a system, after the consistency and non-dimensionalisation process, each design theory module would be called in sequence. This sequence should be also controllable by some kind of Design Control Module. Basically, each program called collects data from the data-base and, after running, stores its results back in the same common data-base. The results of one or more programs may then be called as input for the next one, and so on. Because the design process is so complex the interdependency of parameters is evolutionary and not necessarily sequential. Therefore several loops through the whole set of theories is usually necessary to increment or "mature" the results.

This automisation concept opens way to the use of optimizers, that can perform full sets of various loops of the system, under the control of an optimization control device that would vary parameters according to the strategy of the method deployed starting from some initial chosen design. This would be used in conjunction with manually carried out designs.

When running a system manually, by studying the results of trial parameters, it should become clear to the naval architect what to do to adjust his guesses for better results. These trials also help to identify and correct mistakes in the sequence order of the design theory modules and can be used either as adjusting runs to set the data-base for optimization, prescribing objectives, constraints, etc., or as final adjustments after the optimization process has ended.

There can be some, not entirely obvious pitfalls in using a cyclical system such as this. Besides the difficulties already mentioned concerning consistency and sequencing there remains the danger of cycles that drive up unrealistically certain design parameters, e.g., when increases in size require increases in installed power which further drive up the required displacement, see Andrews[4]. This aspect is discussed further in Chapter 4, where it is shown how it can be avoided.

Another aspect discussed in Chapter 4 is concerned with three-dimensional hull-form, which might seem to be undermined by the cyclical process. Clearly the design control module must be able to allow the form to be distorted or modified automatically. This feature must obviously not restrict the design process and the availability of a suitable process will be seen to be critical to automated ship design.

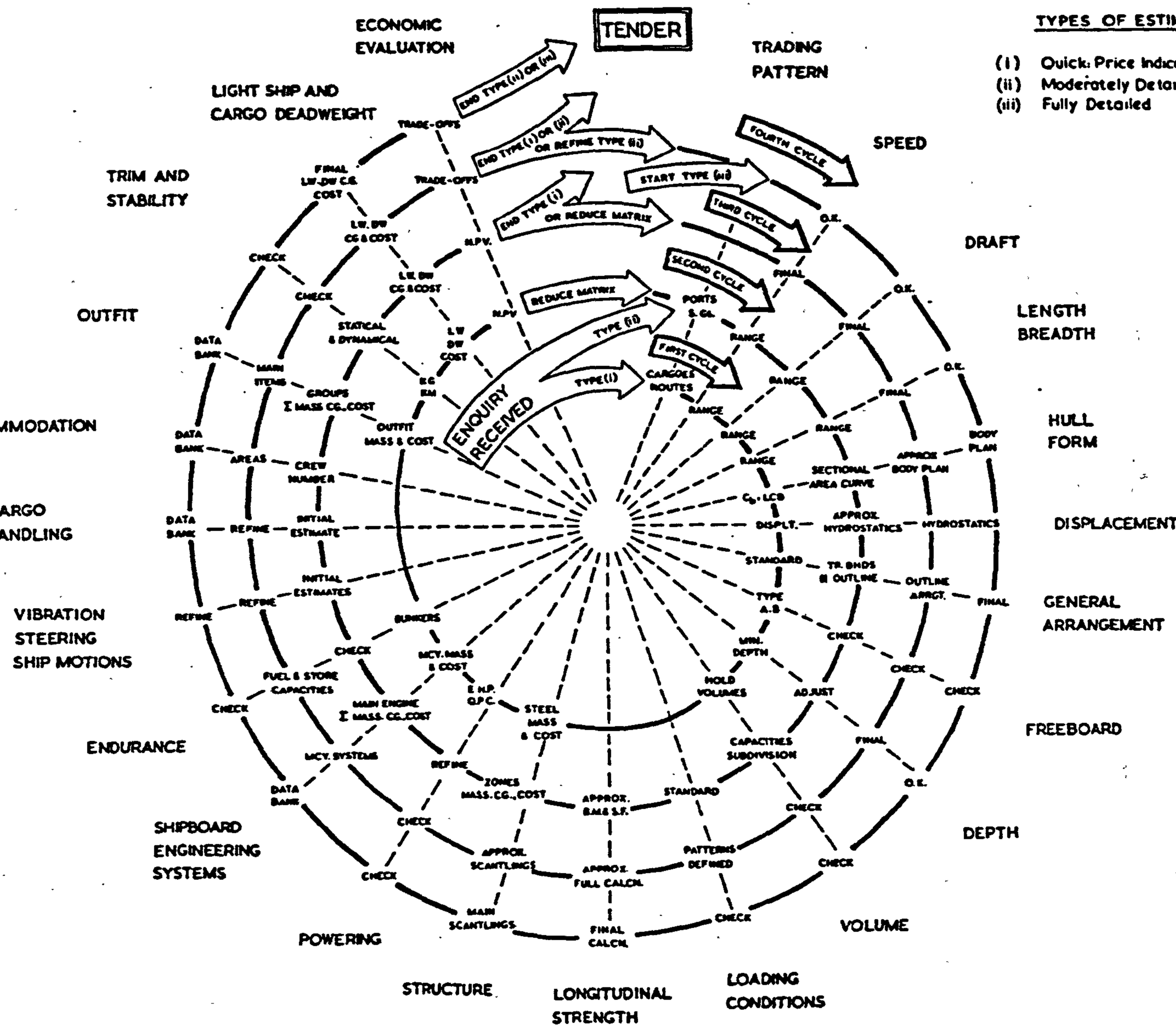

Figure 2.1 - The Design Spiral

3. OPTIMIZERS

3.1 Introduction

Optimization has been widely evaluated in engineering applications of all sorts[23]. The ever increasing use and development of computers has made this technique more advantageous by making it an increasingly automated process, as opposed to laborious manual approaches where only a few configurations were calculated due to lack of time.

Optimization methods have proven to be a useful tool in applications where the problems are of a deterministic nature, where, after all possible considerations, a unique solution must be given in order to allow a physical item to be produced. This does not mean that the calculations must always result in precise values. Rather, it means that regions of good solutions should be found, within an engineering precision, i.e., certain parameters, such as power, displacement, etc., should have determined orders of magnitude, and others, more precise values rounded up to practical values, such as thicknesses, etc.

Optimization is useful in engineering design because of its ability to perform goal oriented tasks; targets can be specified and optimizers deployed so as to drive a design towards desired goals. These are specified numerically, and in order to make such improvements, the requirement of calculating variations in some measure of merit arises. As has already been mentioned, usually the aim is to make certain

parameters as large or as small as possible although conflicts often arise. Of course, once such a mechanism for improvement has been found it can be pursued until exhausted. This is precisely the function of optimization techniques.

Another very important reason for applying optimization techniques is that the type of exploratory searches they perform may be innovative, because they try, by systematically varying parameters, previously untried combinations of ideas. If the outcome of such trials is successful, the result will be captured and improved.

Siddall[23] shows several example areas of engineering where optimization techniques can be successfully applied. These are in structural design, heat exchangers, steam condensers, car disc brakes, a diving submersible, a rolling mill, the suspension of a vehicle, beer blending, etc. In ship design there are several different applications, as has already been mentioned in Chapter 1. In the end what this means is that, if a design problem can be modelled mathematically, then optimization techniques may be successfully applied to gain better results.

3.2 Definitions

Many different definitions are given for optimization. In a general sense optimization is concerned with trying to achieve the best possible result for a given problem, following certain criteria and satisfying any restrictions involved. Siddall's[23] formulation of the

design optimization problem is as follows. Produce a general configuration that can be set in a mathematical representation in the form of equations in which numerical values of the independent variables have not been fixed. Then in order to obtain the best specific configuration, the problem must be tackled in a general way as follows. An optimization or objective function (U) is set up defining the total "goodness" of the design in terms of the independent variables (x_1, x_2,...x_n) that characterise a particular solution and are called a trial vector.

$$U = U\ (x_1,\ x_2,\ \ldots,\ x_n) = \text{maximum or minimum}$$

Equality and inequality constraints (Ψ_i and ϕ_j, respectively) are developed which define feasibility with respect to all restrictions and possible modes of failure.

$$\psi_i(x_1,\ x_2,\ldots x_n) = 0 \qquad i = 1,m$$

$$\phi_j(x_1,\ x_2,\ldots x_n) \geq 0 \qquad j = 1,p$$

Numerical procedures can then be used to adjust the independent variables so that these expressions are satisfied. These procedures are the optimization methods and the process is a goal oriented task.

Optimization methods systematically vary trial vectors, according to their own in-built strategies, and verify if the objective function is changing in the desired direction or not and by what amount, as well as keeping track of any constraint violations and their severity. This

tells the optimizers which parameter to change and in what direction, as well as other controlling features such as step sizes, areas to investigate further, areas to move away from and by what amount, etc.

In the system to be described in Chapter 4, each constraint and variable that forms the constraint and trial vectors is a parameter in the data-base as is the objective function. In the case of the objective function it is also possible to combine several parameters in the form $U = U_1 + U_2 + \ldots + U_n$, where any U_1 can be transformed with a weighing or have an opposite requirement to the others (i.e., be maximised when others are minimised), etc. Using multiple parameters as an objective function represents a "softer" specification than if they were used as constraints, i.e., they have to be as big (or small) as possible rather than not less (or more) than a certain value (or range). The development of suitable functions requires a careful analysis of the aims and compromises the design needs to fulfil.

In order to produce a good system successfully using optimization, some expertise is, of course, required: the problem has to be specified using the right equations and a good balance between them is required; the user must have a good understanding of modes of failure or constraints and has to have those well defined, the sequencing of the routines must be well thought through and the objective function, in the form of one or multiple parameters, must be correctly chosen. If this is done then the optimization process will prove to be a very powerful tool for compromising conflicting features. Moreover, by investigating results and constraint violations, it will give good insight on any

corrective actions to be taken and also for improvements to be made to the design.

3.3 Optimization Techniques

As has already been mentioned, there are many optimization techniques available. Some suit some types of problems better than others. It must be said that optimization in the wider sense does not necessarily mean applying techniques or methods which are mathematical rationalisations to aid in the problem solving process. Furthermore, optimization does not necessarily imply that it is sensible to use a computerised analytical optimization facility to solve problems. Some optimization techniques are, of course, applicable as they were originally conceived: they can be used manually in a calculation in a traditional way, as with linear programming. In some cases this is more economically justifiable. The case of interest here is the application of optimization methods as they have developed to take advantage of the progress in computers, such as speed of calculation, plotting abilities, etc.

Among the many techniques available, to mention just a few that would suit this type of problem, there are the linear programming techniques where methods such as the Simplex can be employed, Random methods, Gradient methods, Direct Search methods, etc. Methods can be linear or non-linear, constrained or unconstrained (in which case one would use Penalty Functions to control constraint violations). For the present work only a few optimization methods have been studied, in number just

enough to demonstrate the ability of the system to use different methods and also to see how the different methods compare.

A package developed by Siddall[23] called OPTIVAR has been used for this purpose. This package contains various optimization methods in a format that enables simple use of a variety of modern techniques. It provides simple means of controlling and manipulating features of each method, and this allows a better comparison between them. These methods may be classified as: (a) constrained non-linear methods; (b) unconstrained non-linear methods combined with penalty functions; (c) single-variable minimisation; (d) linear-programming. Of these, the constrained and unconstrained non-linear methods are relevant to the problems being investigated in this work. The constrained non-linear methods available are :

(1) Minimisation by the method of successive linear approximation (APPROX).

(2) Minimisation by random exploration with shrinkage (RANDOM).

The unconstrained non-linear methods are a group of different strategies all using the same interchangeable penalty functions to deal with constraint violations:

(1) Minimisation by adaptive random search (ADRANS)

(2) Minimisation by Davidson-Fletcher-Powell method (DAVID)

(3) Minimisation by Fletcher's 1972 method (FLETCH)

(4) Minimisation by Jacobson and Oksman method (JO)

(5) Minimisation by Powell's direct search method (PDS)

(6) Minimisation by Hooke and Jeeves direct search (SEEK)

(7) Minimisation by the simplex method (SIMPLX).

The penalty functions available are :

(1) One pass external function (OPTIM 1).

(2) Fiacco-McCormick combined external and internal function (OPTIM 2)

(3) Powell's function (OPTIM 3).

(4) Schuldt's function (OPTIM 5).

All of these methods are briefly described by Siddall[23], where further references to the original works may also be found. For the present work, the two constrained methods were adopted together with one of the unconstrained methods (SEEK) combined with the first two penalty functions (OPTIM 1 and OPTIM 2). These represent four different strategies: APPROX is the quickest method if the function is smooth enough to be approximated to a linear one in the region of investigation. It tests the vertices of the n-dimensional form created from the linearised objective function allowing for planes formed by linearising the various constraints. RANDOM is a random "shotgun" search method; it is very slow, but also potentially the most reliable. If used in its purest form it will always find the optimum and so is

used as a reference for the evaluation of the other strategies (a shrinkage mechanism is invoked by default with this method to narrow the area for searching but this introduces the risk of finding a false optimum). Finally, one of the unconstrained non-linear methods was selected. The Hooke and Jeeves method was chosen since it is both well known and typical of an heuristic approach. It is a direct search strategy that tests variables systematically and then decides the directions and sizes of steps to be taken using various comparisons. The penalty functions employed are basically of two kinds: a one pass external function, which penalises the function only when the constraints are violated, and then extremely strongly; or multiple pass functions which penalise the function both inside the feasible region as the constraints are being approached as well as in the infeasible area. The Fiacco-McCormick, Powell and Schuldt functions are multiple pass functions having slightly different formulations, but all rely on penalties that become increasingly severe as repeated optimizations are carried out. These latter types of penalty functions are useful when the function being optimized cannot cope with variables that lead to infeasible results, making the system 'crash'. However, since they are mild in action and try to avoid going near infeasible regions of the function where optima often lie, this approach tends to be slower than the first. For robust systems, i.e., systems that can cope with calculations in infeasible areas, the one pass external function proves to be the quickest. Appendix A gives a more detailed explanation of the methods and penalty functions used here.

These four approaches have various advantages and disadvantages. RANDOM, as has already been mentioned, may be very slow, particularly if the feasible region is rather restricted and there will then be a high risk of a large number of infeasible trials. On the other hand, it does not "hang up" on false or local optima. For this reason, it is a good method for checking the results of other approaches. Care must be taken to ensure that the shrinkage mechanism used does not reject the true optimum. With APPROX, if the function and boundaries formed by constraints are not amenable to linear approximation, although finding the optimal region quickly, it will have great difficulty in converging. Testing of APPROX has shown that even functions such as the one studied in this work (see Chapter 4) can sometimes be successfully approximated as linear over the small ranges required. However, highly constrained functions or those containing singularities are not handled well. With SEEK, the Hooke and Jeeves method, the main difficulty arises in highly constrained problems. The search can get stuck on constraints because of the fixed orientation of its search co-ordinates. If the contour lines and the inequality constraint lines happen to have certain orientations, and the search approaches from particular directions, the search may become stalled. A new starting point may well alleviate the problem. Some types of 'soft' penalty functions may also be helpful in extending the search. A local random search may help the search to jump out of trouble and this is included by default with the one pass penalty function.

Besides easing the choice of the optimization method and penalty function to use, the OPTIVAR package also allows the user sophisticated

control of the optimization process by modifying or "tuning" some features. The use of this facility is described in Section 6.3 - see also sections 4.2.5 and 6.1. There are a great number of these features, some common for a few methods, others specific to just one. These may control the maximum number of runs or search cycles or linearised steps, number of shotgun searches, limits on step lengths, shrinkage factor, etc.

3.4 Limitations

Optimization is very often criticised for being a time consuming and unreliable technique. It is very common to notice that design optimization methods are not really adopted as common daily tools in design offices. The following problems are clearly apparent :

(1) Lack of Familiarity - the feeling of uncertainty the user has when not being able to visualise the process, which is almost always complex, multidimensional and highly constrained.

(2) Complexity - there is the fear of getting results that represent local or false optima and it is natural that the user has little notion of how probable or improbable this can be, because the shape and behaviour of this complex function, which is the whole design, is unknown.

(3) Slow Speed - to go around a great number of design runs (which has usually been performed[32] using Random methods) creates the problem

of making the technique extremely time consuming. Engineers trust their knowledge, their experience and insight and methods that investigate improbable regions or give detailed answers in good but too wide regions may look useless to them.

(4) Poor Reliability - when more rational and sophisticated search methods are applied to give quicker results, it is found they are liable to get stuck in constrained areas (which in engineering are the most likely ones to have the optimal compromise).

(5) Cycling abilities - another problem of optimization is the fact that such a process requires automation, i.e., so that it can be run many times in cycles allowing for changes in values of some parameters. Most complex design processes require work that seems to be impossible to put into a computer program. The way around this is obviously to develop programs that will allow automation. To overcome this, optimization design systems often simplify their tasks to parametric design studies. Also, it is common for optimization processes to spend time trying to reach an optimum at a precision which is not justified for engineering studies (but may be necessary in other areas).

Another aspect of the occurrence of local optima has to do with variable range. The setting of reasonable boundaries is a necessary requirement of the design process and therefore this should not be used as an argument against optimization.

3.5 The Mapping Strategy

Once a design problem is identified, functions in the form of programs are defined, the goals to be achieved are known, etc., the obvious next step might seem to be to allow the optimization methods complete freedom over the variables of interest and then to set the system running. This might seem ideal but it is not practical. It is precisely because of this sort of desire that users often end up concluding that optimization is unreliable. The limitations already mentioned must be studied beforehand. Complex systems take a long time to run. The one to be presented in Chapter 4 requires some 90 seconds for a single ship design (or loop) on the machines used. Therefore, runs of a thousand loops require around 25 hours to carry out and such numbers of loops are commonly required when optimizing over many dimensioned spaces. Secondly, it is necessary to verify that the optimum found by an optimizer is the true global minimum. Some problems have, as has already been mentioned, local sub-optimal 'peaks' or local optima and others constraints that can 'stall' optimizers. Consequently, a more sophisticated strategy is required to get the best from the optimization process. The following strategy is in accordance with and takes advantage of the facilities developed in the system to be described in Chapter 4 and it is developed in the form of a knowledge base in Chapter 6, which gives advice on how to better use optimization methods to obtain trustworthy and fast optimal designs.

(1) First, select an objective function that is considered a reasonable measure of merit for the proposed design. This is

quite likely to contain terms representing numerous desirable characteristics in the final design. This, in itself, is a worthwhile design discipline.

(2) Next select as many constraints on a realistic and workable design as can be found (in ship design these would be stability, strength, capacity, etc.).

(3) The constraints are then tested by supplying an initial design (normally an educated guess or a design generated by defaults) in a manual design process. If too many constraints have been selected or they have been drawn too tightly this will prove difficult (or perhaps impossible) and this may lead to a revision of these quantities. It also allows the designer to carry out an interactive design session in the classical manner.

(4) Next choose a set of design variables that are thought to have a significant influence on the objective function, to form the trial vector. The number of variables chosen determines the dimension of the function to be optimized and since it is desirable, during preliminary work, to produce a 3-D contour mapping of the objective function, the list of variables is next reduced to two. These are chosen to be the most dominant ones for the problem, usually by trial-and-error. Their extreme upper and lower limits should also be established, and this reduction allows the generation of a contour map of the objective function for the selected variables, subject to the given constraints. This map

shows the type of function involved, indicating its behaviour as the variables change, revealing whether the parameters chosen do in fact control the objective function and also highlighting which constraints are active. If there is any local or false optimum, as well as saddle points within the function, this should be clearly seen. It also allows subsequent optimization studies to be kept away from testing unpredictable combinations of variables that might otherwise cause difficulties.

(5) The next step could be considered as producing cross-sections of a multidimensional hyper-space. From the first mapping fix one or two pairs of the dominant parameters at points of interest. Then produce 3-D contour mappings of two of the other parameters that were left out (fixed) at the first mapping. These variations show if any or both of the dominant parameters are worth mapping against a third one, again testing interdependent changes to the function. Sometimes it may also be useful to produce a 3-D contour map similar to the first, but at an opposite limit of the third most dominant parameter. This will show how the first mapping's "landscape" will be distorted and this can give insight into the inter-relations of the three strongest variables. This methodology gives good insight into the multidimensional problem, but it is by no means enough to predict the multidimensional optimization path and all the implications of the process.

(6) A starting design for optimization, which represents reliable and widely used practice[34,23] can be then selected. This point can be

a first guess or a default design, but it is perhaps better to use the best possible position seen in the first mapping for the two dimensional problem. If there are regions of local or false optima or saddles, the starting point should lie as far as possible from them. It is also advisable not to use a starting point too close to the visual constrained optimum, rather it should be located in a feasible area away from this optimum. This will tend to avoid non-convergencies or hang ups, allowing the methods to gain some "momentum", i.e., to exercise a couple of systematic variations of trial vectors in order to get a notion of the right optimization direction before decreasing their step sizes. It also confirms that the methods are working and have not converged by chance.

(7) For simplicity, optimize for the first two, most dominant variables, so that the optimization process can be plotted on the contour map and different optimization strategies compared for the objective function under examination. This two dimensional optimization practice prior to the n-dimensional one also helps to get the design process closer to the optimum more quickly and in a more controllable way, but this will only be true if the first two variables really are dominant. During this process the system should allow, as will be seen in Chapter 4, the user to monitor the optimization by displaying the status of the process at pre-defined numbers of cycles, or intermittently using an interrupt facility. When the various optimizers finish, a status message should be generated and if a true optimum has not been

found, an explanation given. The reasons for failure will vary; they can arise because the method chosen is not capable of further minimisation, or perhaps is unable to cope with an infeasible starting point, or simply that the amount of computer time specified by the user was too little to allow convergence.

(8) Finally, once this process has been fully studied, proceed to the n-dimensional problem that arises when carrying out the full optimization, and for which maps cannot be produced.

At this point some confidence in the final answer may be expressed without resorting to exhaustive testing. However, further manual manipulation of the design may be informative for the user, either in verifying the qualities of the suggested design or in re-defining the optimization problem, whereupon the above strategy can be re-entered. As will be mentioned in Chapter 4, failures during optimization can also arise when very wide limit settings are given for the trial vector, that allow the optimization to reach unpredicted singularities. These are usually caused by the optimizer selecting inconsistent parameters. In the case of ship design such inconsistencies could arise from inconsistent hull-form parameters, such as extreme flare combined with a full midships section. For these reasons, monitoring and interaction are very important during this combined manual and automatic design process, because suitable direction to the design process can then be given. Usually, interleaved sequences of interactions and optimizations are found to be most appropriate when developing a design.

Once this mapping strategy has been applied combined with optimization, the questions of precision in the final result as well as the recognition of a true optimum remain. Good engineering solutions reflecting compromises between the various conflicting requirements are usually self evident. They do not really need to be particularly precise. This is akin to the case of an airline traveller who would like to arrive at his home address, rather than in the City's Airport. Conversely, in many cases engineering judgement is enough to realise that a particular solution may represent arrival at the wrong City. This judgmental ability helps the designer, but mapping strategies supported by some expert rule-based advice on how to use optimizers can help a great deal in avoiding this.

3.6 Summary

Optimization is a technique that clearly has many advantages. If well applied it can overcome most of the difficulties that are often pointed out, leaving the achievements it can provide. This is possible now due to the ever increasing speeds and graphical capabilities of computers, making what was seen as a tool of great potential in the past, perfectly feasible for everyday use.

Once the user gets involved in the process it is found that even complex non-linear functions which are heavily constrained have certain quite controllable patterns of behaviour and many of the singularities identified are clearly defined or detectable in the design theory modules. They usually arise from discontinuous functions which the

users are generally aware of. It is also noticeable that it is not uncommon to have regions of false or local optima that represent infeasible designs rather than singularities. Proper monitoring devices in the design theory programs can easily tackle such problems and the use of mappings helps further.

It must be said that mapping strategies for visualisation are crucial for a successful use of optimization. Once the function can be visualised by a combination of mappings, regions of false or local optima can be detected, hang ups of optimizers can be tackled and solutions can be found within engineering precision.

A mapping strategy combined with rule-based expert advice should help to save a great amount of computing time, because the optimizers are then controlled to first perform wide ranging searches over worthwhile areas and then to confine these studies to smaller but selected regions for faster objective searches. By visualising a restricted area the user may decide that he can sort out a hang up by making a local shotgun random search in a similar way to the strategy used by SEEK with OPTIM 1. These improvements to the optimization process will be seen in Section 6.4, where a rule-base (Section 6.3) developed for controlling optimization is applied to the system described in the next chapter.

Nevertheless, the use of such tools is not foolproof and, as it should be noted, still requires expertise to use well. The system to which the technique is applied must be well structured, the user may be inexperienced in optimization, but must understand the problem involved,

the specification, limitations and possible modes of failure. Using a mapping strategy that graphical abilities make possible could be considered to be very important to optimization in design, but good analysis and judgement capabilities are still required. In summary, it seems that there are three areas to be catered for in order to make optimization techniques capable of being a powerful tool in complex designs:

(1) a rational structuring of the routines that compose the design disciplines involved, (2) the graphical ability to allow the mapping strategy to aid optimization and (3) additionally some expert rule-based advice for objectively controlling the technique.

4. APPLICATION OF OPTIMIZATION TO SHIP CONCEPT DESIGN - AN EXAMPLE

The system to be described here has been developed using the UNIX operating system on SUN 3/50 workstations. It is written in ANSI 77 Fortran, and requires 1.5 megabytes of memory for processing. The system is fully portable, and can run on any UNIX based computer supporting ANSI 77 Fortran and having the required memory. Graphical output makes use of the widely known GINO[51] package for 3-D views, body plans and water plans and Simpleplot[52] for contour mapping.

4.1 The Design System Structure

The structure of the design system is illustrated in Figure 4.1. The basic idea has been to develop a flexible framework together with a small number of components, covering just enough topics to allow satisfactory designs to be performed whilst evaluating the techniques employed, following the specifications of Section 2.4. The system allows the integration of the various stages of design (or areas of knowledge) in naval architecture in the form of routines called Design Theory Modules. The presence of certain modules, even in crude form, is necessary to ensure a balance between the competing aspects of naval architecture, enabling the study of realistic problems.

The system is structured to provide flexibility, allowing any design theory module to be altered or replaced. This is possible because each design theory module is called by a central Design Control Module and

has a standardised interface in a pre-programmable sequence. The system is also structured to use a common data-base, controlled by a Data-base Handler, which also forms the interface with the user. The opening sequence of each theory module calls all necessary variables to run that routine from the data-base and the closing sequence sets the results or modified variables back to the same data-base. These results can then be used as input to subsequent modules, called later in the design process or even earlier on in the design sequence of a subsequent loop in the spiral. This allows the automation of the design process enabling it to modify the design in multiple design cycles. Such loops can be controlled by the user, or, more normally, by the optimizer which can be invoked by the user through the data-base handler, when he is satisfied with an initial definition. As will be seen in later Sections, default ships can also be used to form a starting point. The data-base handler prevents theory modules looking for undefined variables by allowing the user to specify them, without interrupting the design process.

Besides the Design Control Module, the Design Theory Modules, the Optimizer and the Data-base Handler, there are some auxiliary routines for plotting and drawing already mentioned. The design theory modules currently available cater for hull-form creation, hydrostatics and stability curve calculations, stability assessment, enclosed volume estimation, light weight estimation and resistance calculation.

4.1.1 The Data-Base Handler (GENDAT)

The GENDAT data-base handler has been developed by Keane[53] in the form of a library of Fortran functions that may be readily incorporated into new programs. The primary function of this package is to provide the means of controlling and accessing the design data-base, a formatted file holding all data in a sequential fashion, in which a ship is parametrically defined. The requirement for flexibility in this capability means that, in addition to providing the means for storing and retrieving data, it also provides a powerful, program definable, user interface.

When the design control module is run, the interface is entered and it initiates a menu driven interactive process with the user. It is from this level that all subsequent routines are accessed and controlled. Using the package a ship description file can be recalled from disk, or calculated results sent back. It additionally incorporates features that allow the input of values for the various parameters of interest and the selection of suitable constraints and objectives together with the parameters to be varied during optimization. It also allows the creation of formatted output files for subsequent printing. In order to describe the vessel, the design data-base maintained by this system contains the names of parameters, their values, a type description (defining if the parameter is the objective function, a constraint, an optimization variable; if it is fixed; or even to what theoretical group it belongs, e.g., hull-form, stability, resistance), their units, definitions, etc. Any of these can be accessed individually or in

groups; they can be created, added to or modified, enabling the user to have great flexibility and ease of control. When the design process or even individual pieces of theory are run, it is at this level that the head and tail sections of the design theory modules interact with the data-base. The system may be run in one of two modes, manual or automatic, directly from this interface.

4.1.2 The Design Control Module (CONSST)

When the user starts the design process, control passes to the design control module which forms the heart of the system. It is this routine which selects the order and choice of design theory modules and auxiliary routines. Within it, ship parameters from the data-base are made mutually consistent and form parameters non-dimensionalised, using a rule-based structure. These features allow parameters to be reduced to a numerically consistent form when initial data are given in an inconsistent way or after one or more variables have been modified. During this process extraneous data are overwritten, following a preprogrammed hierarchy.

Currently, in order to start the process, the following parameters must be given: Ⓜ (length to displaced volume ratio); B_{WL}/T (beam to draught ratio); C_P (prismatic coefficient); C_B (block coefficient); ∇ (displaced volume) and D/T (depth to draught ratio), the last three being fixed for each run. Hierarchical rules will then generate L_{WL} (waterline length), T (moulded draught), B_{WL}(waterline beam), D (depth) and C_x (maximum section area coefficient), as follows:

$$L_{WL} = (m) \times \nabla^{1/3}$$

$$T = \sqrt{\nabla/(L_{WL} \times B_{WL}/T \times C_B)}$$

$$B_{WL} = T \times B_{WL}/T$$

$$D = T \times D/T$$

$$C_X = C_B/C_P$$

Of course, like any other structure, this hierarchy could be easily changed and designs could then be made for different combinations of the main characteristics, like length and beam, say. Usually, in a given design problem, some of the key parameters will be fixed, or at least based on previous experience; all that is required here is that the various parameters be made compatible and it is this aspect that the structure addresses. A very useful capability of the system would be to have such different combinations of principal parameters made consistent without having to change the code of CONSST. This problem is considered in Chapter 5 where the use of an expert system shell makes this easily programmable, giving insight of how to incorporate such a structure into normal Fortran code.

The Design Control Module can be used in one of two modes, both accessing the same naval architecture theory and a common data-base, i.e.,

(1) Fully manual mode - all design decisions are taken by the user and individual theory modules are selected manually to support this process. This mode is similar to many existing naval architecture design tools.

(2) Automated mode with optimization - a few key decisions are taken by the designer to select the required goal and theories to be used. Optimization techniques are then employed to drive the design towards the desired goal with the theory modules being deployed by the optimizer only and when desired.

Since both modes use common theory and data, either can be used to reach the same final design and this provides an important check on any solutions suggested by the optimizers.

In the automated mode the optimization methods, when called, deal with the objective function and constraints separately, according to the chosen strategy. In order to save run time the theory modules are separated into two groups OPTFUN and OPTCON, dealing with these two aspects respectively. Currently OPTFUN runs the hull-form creation, hydrostatic characteristics, enclosed volume and light weight estimations, the loading condition hydrostatic characteristics and the resistance calculation, while OPTCON, which groups restrictive theory (for constraints), runs the hull-form creation, hydrostatic characteristics, enclosed volume and light weight estimations, the cross curves and GZ curves of stability, and the stability accessment criteria. Since the user is free to set any parameter or group of parameters to be a constraint or the objective function, it is necessary to be able to change the positioning of the routine that contains relevant calculations. Again a relatively simple, rule based structure could be constructed to obviate this requirement.

In order to maintain the cyclic ability of the system in a form consistent with the detailed calculations carried out at the early stages of design, a rule-based structure, using a non-dimensionalising process for hull-form parameters, was inserted in this routine. It is desirable during optimization or consecutive manual designs, that independent variations of one or more form parameters do not make the hull-form creation impossible. By using non-dimensional hull-form parameters, these parameters can be expanded into a dimensionally consistent set of parameters at each run.

This non-dimensionalisation and subsequent dimensionalisation expansion is carried out using a coherent set of parameters: the overall breadth is scaled by the waterline breadth; the lengths of the parallel body and keel by the waterline length; the flares and rise of floor by the waterline breadth and moulded draught; the rakes by the waterline length and moulded draught and the half angle of entrance by the waterline length and breadth. This process can be accessed in any of the following ways:

(1) The user can specify a type ship for the non-dimensional values - in this case default non-dimensional quantities for that type are used.

(2) The user can specify dimensional parameters, for instance, from a series or a known ship.

(3) The user can specify non-dimensional parameters.

(4) The user can combine any of the above. For instance, he can select a type ship and decide to override some of the parameters in dimensional form and others in non-dimensional form.

In order to allow for this process, a rule based structure was designed to recognise the type of the variable prescribed in the data-base. If they are ordinary variables (e.g., hull-form, etc.), and the ship type is given, the non-dimensional values of the defaults for the given type will be taken and will overwrite those in the data-base. If any non-dimensional parameters in the data-base are prescribed as FIXED (or chosen to be an optimization variable, OPT-VAR) they will be maintained. In either case the dimensional equivalents are formed from the non-dimensional terms. If any dimensional parameter is prescribed as FIXED (or OPT-VAR), it is maintained and the corresponding non-dimensional quantity created. This procedure allows the system to use hull-form parameters as design variables in addition to the other main characteristics.

4.1.3 The Optimizer (OPTCTL)

A number of different optimization strategies are available within the system, drawn from the OPTIVAR package of Siddall[23] (Chapter 3). When running the system in its automatic mode, a linking routine called OPTCTL provides data from the data-base to OPTIVAR in a suitable form. This suite contains seven main methods of unconstrained non-linear optimization with four associated penalty functions, together with two constrained non-linear methods, see Section 3.3. As discussed in

Section 3.2, the user chooses the objective function, constraint vector and trial vector, by modifying parameter types within the data-base handler. It is the trial vector that is changed at the beginning of each step of an optimization search. Within the OPTIVAR package the optimizers all try to minimise a single parameter defined to be the objective function. This can be resistance, for example, or any parameter or combination of parameters, forming a weighed sum, where minimisation is desirable. If any or some of the parameters are to be maximised, all that is required is to give the negative to be minimised. Of course, the parameter chosen must be calculated as part of OPTFUN.

To achieve an optimum the optimizer modifies variables selected by the user to form the trial vector (e.g., C_P, breadth to draught ratio, length, volume, etc.), according to the strategy in use. These variations are carried out within pre-established upper and lower limits set in the data-base. When seeking constrained optima, which is the normal case, the optimizer tries additionally to fulfil separate constraint requirements (on parameters such as the enclosed volume, stability criteria, etc.), and keep records of constraint violations. As an example, let the trial vector be formed by, say, the length and the beam of the waterline (L_{WL},B_{WL}), the constraints from stability safety factors (SF(i), i = 1, m), and the objective function to be minimised the total resistance (R_T); then L_{WL} and B_{WL} are set as type optimization variable, SF(i) as type constraint, R_T set as objective function and values are given to L-L_{WL}, U-L_{WL}, L-B_{WL}, U-B_{WL}, L-SF(i), U-R_T, where L- denotes lower limit, U- denotes upper limit. For the trial vectors and objective functions these limits are used to keep the

search restricted to reasonable, wide but not absurd limits for the sake of saving time. For the constraints, the limits are actually those limits beyond which the constraint would be violated.

One additional feature that has been incorporated in the optimizer package concerns the evaluation of the objective function for values of the trial vector where the calculation breaks down (possibly due to singularities within the function, geometrical inconsistencies, etc.). When this occurs the system uses the previously calculated objective function value arbitrarily incremented by 10^{10}. Constraint functions that cannot be calculated are decremented by 10^{10}. These features tend to drive the optimizers away from areas where the system is unable to calculate data, allowing the optimization to continue where it would otherwise 'crash'. However, as is noted later, even this technique is not foolproof in cases where a particular optimizer relies on consistent data over the entire region being studied.

Finally, the optimization process can be interrupted at any time the user feels it necessary, so that corrective action can be taken. The system also prints out, at each loop, the loop number of OPTFUN and OPTCON, the values taken by the trial vector, the constraints and the objective function to allow monitoring of the process.

4.1.4 The Design Theory Modules

The design theory modules used by the system form a basic set of naval architectural routines. They are sufficient to illustrate the work done

here although they could easily be expanded. They currently comprise:

HULLCR and HULLDR for hull form creation and drawing.

STABCV for hydrostatics, load condition draughts and trims and GZ curve generation.

SPACE for enclosed volume estimation.

WEIGHT for load condition displacement and CG estimation.

RESHLT for resistance calculation using Holtrop and Mannen's[9,10,11,12,13] power prediction method.

STCIMO for stability evaluation using relevant parts of the IMO A-287 criteria (with increased severity when applied to warships).

4.1.4.1 Three Dimensional Hull-Form Creation and Drawing (HULLCR-HULLDR)

This routine, derived from the method developed by Keane[22], defines a simple hull-form from a set of nineteen parameters. Once the hull-form is defined, the system places its off-sets into the data-base. This forms an initialisation process, the results of which allow more detailed calculations to be performed. It also serves as a powerful type of constraint, i.e., the design will only proceed if a three-dimensional hull-form for the specified main characteristics is viable, thus avoiding combinations of parameters that could not represent a hull-form and would otherwise waste calculation time.

If an inconsistent set of parameters has been specified, suitable steps are taken, that will halt the manual process, will force the optimizers to modify the trial vectors, or will place boundaries, representing inconsistent hull-forms, on contour maps produced by the auxiliary routines.

The hull design method used by HULLCR is completely mathematical, requiring no further input once the initial parameters are chosen. For the below-water form it uses a variation on the 'Lewis Form' method allowing for flare and rise of floor using very few section parameters. The above-water form is given as a quadratic function, tangential to the below water form at the water-line. The water-planes and upper deck are defined as cubic polynomials in longitudinal position forward of the parallel body and quadratic aft of it.

The advantage of using this method is that hull forms can be defined extremely rapidly, in full, from few parameters, discarding complex hull fairing processes. This method of definition for sections, water-planes and profile allows great flexibility in modification, scaling and distortion of the form. These capabilities are fundamental to an optimization system, where a great many loops through a design may be required, ruling out manual hull-form manipulation methods. Of course, the method has limitations when dealing with certain types of ships, (i.e., chined ones and some extreme forms) but most types of monohulls can be handled. This can be considered a limitation to design creativity, but this represents a compromise between detailed design in the early stages and complete hull-form flexibility that has to be

carefully balanced. The HULLCR routine can be changed, by-passed or even substituted, as is the philosophy for all design theory modules in the system provided some other method for generating the hull offsets is available.

Using this method to define a hull-form, the following nineteen parameters are required:

(1) Waterline Length (L_{WL})

(2) Waterline Beam (B_{WL})

(3) Draught (T)

(4) Depth (D)

(5) Block Coefficient (C_B)

(6) Maximum Section Area Coefficient (C_x)

(7) Overall Beam at the position of Maximum Section (B_{OA})

(8) Length of Keel (L_{KEEL})

(9) Length of the Parallel Middle Body (L_{MID})

(10) Non-dimensional Position of Maximum Beam (L'_x)

(11) Ratio of Waterline Transom Width to Waterline Beam (BT_{WL}/B_{WL})

(12) Ratio of Overall Transom Width to Overall Beam (BT_{OA}/B_{OA})

(13) Rise of Floor Angle (ROF)

(14) Waterline Flare at Maximum Beam ($FLARE_x$)

(15) Waterline Flare at the Aft Perpendicular ($FLARE_A$)

(16) Half Angle of Water-plane-Entrance (i_E)

(17) Upper Deck Half Angle of Water-plane Entrance ($i_{E\ UD}$)

(18) Forward Rake Angle ($RAKE_F$)

(19) Aft Rake Angle ($RAKE_A$)

As has been mentioned earlier currently only (m), C_P, B_{WL}/T, D/T, C_B, ∇ (displaced volume) and a ship type need be input to define the hull form. From these primary parameters, L_{WL}, B_{WL}, T, D and C_x are generated and the type specified used to indicate a set of non-dimensional defaults for the thirteen remaining parameters. As has been described, if desired the user may input all of these thirteen parameters or just some of them, allowing the designer to impose a chosen style on the final form. The defaults provide convenience by using type-ship data to provide a useful starting point for design. The default parameters currently available, in non-dimensionalised form, for bulk carriers and frigates could easily be supplemented by previous designs produced by the user. The number of types could also be expanded. Geosyms of the actual default ships are generated if all the input data are derived from defaults. Otherwise, the user starts with a typical ship of the chosen type that can then be modified during the manual design or optimization. (Note that this does not imply that the program is restricted to simple distortions or scaling and each of the default ships can be continuously altered to reach any of the others.) Of course in practical design work it is likely that the main particulars would be given, and the non-dimensional versions derived from them. This is possible with the system, but when examining wide ranges of parameters it is found simpler to start with their non-dimensional equivalents.

The HULLDR routine allows the hulls created by HULLCR to be plotted using the GINO package. Body-plans, water-planes and 3-D perspective views can all be generated using this routine.

4.1.4.2 Hydrostatic and Stability Calculations (STABCV)

This routine is a general purpose ship stability program that uses polar co-ordinates with 5° variations for 21 transverse sections, defining the hull-form using appropriate integrating multipliers and levers and allowing for the inclusion of appendages. It can calculate hydrostatics, cross curves of statical stability (at constant trim) and load condition GM and GZ curves (allowing for trim induced by heeling). Currently this module is used for three purposes; first the hydrostatic characteristics at a specified design draught and level trim are calculated and used as input by subsequent modules such as SPACE and WEIGHT. Secondly, having established various load condition displacements and centres of gravity using WEIGHT, it is used to establish condition draughts and trims for use in powering calculations, etc. Finally, the GZ curves for intact stability are used in the application of the stability criteria, usually as constraints during the design process.

This program is used in its various different modes at different stages in the design. The generation of the hydrostatic characteristics from a design draught takes the off-sets generated by HULLCR. In the manual mode, these can be used to produce hydrostatic tables while in the automatic mode the data-base is filled with the hydrostatic data which is used by any subsequent routine where it is required. These characteristics, as used by WEIGHT, allow for the light weight estimation, from which the loading conditions are used to generate condition draughts. From these condition draughts, the loading condition hydrostatic characteristics are generated, also as tables, for various

performance calculations - in the current case, resistance. This sequence can be carried out manually or through OPTFUN. The cross curves and GZ curves of stability are generated manually or by OPTCON, for the condition(s) to be tested by the stability criteria. The GZ values for each heeling angle can be tabulated and can be accessed, like all others, for plotting by Simpleplot. The program also allows for studies of symmetrical flooding and damage stability analysis, but this area is not used by the system at present.

4.1.4.3 Enclosed Volume Estimation (SPACE)

This routine is still an embryo of a 'volumes and areas' routine for concept design. Currently, it only calculates the enclosed hull volume from the hydrostatic data and a given (or default) superstructure size. It is used to ensure the availability of sufficient internal space for the design mission (deadweight, payload, etc.). It is currently crude but adequate for the purpose, ensuring against designs with insufficient payload capacity. Further developments would address the positioning of decks, bulkheads, machinery spaces and different densities of compartments, free surface effects, etc.

4.1.4.4 Lightship Weight, Condition Displacement and CG Estimates (WEIGHT)

This routine estimates the light weights and vertical centres of gravity, carrying out assessments for the number of condition displacements desired by the user. The light weight calculations are

based on Watson and Gilfillan's[7] work for hull, outfit and machinery weights. The hull weight (steel weight, W_s) uses the numeral parameter E proposed there and the correction for C_B as well as a correction factor K for bulk carriers and passenger ships. This E parameter was used because of its usefulness in being applicable to a wide range of ship types. For the outfit weight the suggested curves were used for bulk carriers and passenger ships. For the machinery weight the equation provided was used, being based on the Maximum Continuous Rating (MCR) and the number of Revolutions Per Minute (RPM) of the main engines.

For the vertical centre of gravity a simple equation was derived as follows,

$$KG = [(0.3\ hs_1 + D)\ W_{OUT} + 0.65\ D\ W_S + 0.4\ D\ W_{MACH}]/W_{LGT}$$

where:

KG - vertical centre of gravity from keel (m)

hs_1 - height of full width erections (m)

D - depth (moulded) (m)

W_{OUT} - outfit weight (tonnes)

W_s - steel weight (tonnes)

W_{MACH} - machinery weight (tonnes)

W_{LGT} - lightship weight = 1.02 (W_s + W_{OUT} + W_{MACH}) (tonnes)

This formula was derived from several bulk carriers and passenger ships. Applied to a frigate it gives results just acceptable for the present purposes. The reasons for using this formulation for weights and KG lie

in the lack of better data, and the need for a general formulation (for different types of ship) to enable the construction of stability curves and their analysis together with resistance calculations.

Note that, as is common in commercial ship design practice, the hull design condition is for a draught and trim that may, or may not, coincide with a particular operating condition; that is, the lightship weight is calculated from a design draught and the loading conditions are applied afterwards. This differs from normal warship practice where the hull-form is designed at the deep displacement which is also the principal operating condition. This method can be adopted here by setting the hull design displacement to be equal to a calculated condition displacement at the end of every loop of the design procedure. However, such an approach can cause upward spirals in displacement and is not necessary when a detailed hull-form is available allowing several conditions to be analysed with equal accuracy. If some requirements on length, displacement or freeboard, etc. are to be met for a given condition for the overall design, this is more simply and correctly handled as a constraint during the optimization process.

A similar problem may occur in relation to the lightship weight and resistance calculations. The lightship weight is estimated using the design draught and a nominal brake horsepower (PB) that must be estimated (since the actual resistance is not yet established). The resistance calculations are then carried out using the loading condition characteristics derived from the previous estimate, i.e., the condition displacement formed from the lightship weight with loading condition

added. If the brake horsepower derived from the resistance calculation is fed back for the next lightweight estimation, an upward spiral in displacement may again be caused and, of course, this is not a good way to design ships, it being merely a consequence of the cyclic process.

However, if the machinery weight is not updated each loop, care must be taken that sufficient installed power is available for the design mission. The sequence of design theory modules must therefore be handled with care and design runs carried out before a full optimization is made.

4.1.4.5 Resistance Calculation (RESHLT)

This routine is based on Holtrop and Mannen's power prediction method[9,10,11,12,13] and is fully in accordance with that method, except that it currently does not include the propulsion analysis. The method is a statistical one which calculates total resistance from the addition of the following components, ignoring any interaction between them,

$$R_T = (1 + K_1)\, R_F + R_{APP} + R_W + R_B + R_{TR} + R_A$$

where R_F is the frictional (ITTC) resistance, $(1 + K_1)$ the form factor for the hull, R_{APP} the appendages resistance, R_W the wave resistance, R_B the additional pressure resistance of a bulbous bow near the surface, R_{TR} the additional pressure resistance due to transom immersion and R_A

the model ship correlation allowance. Appendages, bulbous bows, bow thrusters, etc., and different stern shapes can be specified by the user or default values used instead.

The choice of Holtrop and Mannen's method is based on the fact that it is acceptably accurate for concept design purposes and covers a wide range of types and sizes of ships. In a system such as this it is more important to have a formulation which is very wide ranging than one that is accurate for any particular type of ship involved, because this allows the process to search for configurations beyond the established ranges, allowing for creativity.

In the paper "A Statistical Re-analysis of Resistance and Propulsion Data"[13], the allowable ranges of parameters used are extended beyond those originally proposed[12] to enable the inclusion of wider ranges of L^3/V, higher Froude numbers (above 0.5), but these limits are not clear and so caution must be exercised near the limits. Possible singularities that may arise in the method due to inconsistent or extreme combinations of C_P and longitudinal centre of buoyancy are rejected by the routine HULLCR, since they represent impossible forms. In the normal design process, where the naval architect calculates the resistance for a range of speeds and designs, these singularities can remain unseen.

4.1.4.6 Stability Criteria Verification (STCIMO)

This routine was developed to check intact stability by applying relevant parts of the IMO A-287 criteria. The aim was to have in the system the ability to exercise assessments dealing with stability curves. Specifically, the GZ curves and initial GM calculated by STABCV for each condition displacement are verified according to:

(1) Initial GM ≥ 0.15m.

(2) Area below GZ curve from 0 to 30° ≥ 0.55m rad.

(3) Area below GZ curve from 30 to 40° ≥ 0.03 m rad.

(4) Area below GZ curve from 0 to 40° ≥ 0.09 m rad.

(5) Angle of maximum GZ ≥ 30°.

(6) Maximum GZ ≥ 0.2m.

Safety factors are generated in the routine in the form:

$$SF(i) = ((STCR_C * 100/STCR_R) - 100)$$

where

$SF(i)$: stability safety factors, i = 1,6

$STCR_C$: stability criteria calculated

$STCR_R$: stability criteria required

These are regarded as stability constraints in the optimization process and must have positive values. (L-SF(i) = 0.0, the lower limit of SF(i), is set as zero in the data-base for each of the criteria.)

Note that these criteria are intended for commercial vessels and those for warships are more severe. To allow for this variation, the percentage exceedances of the criteria, as used during optimization, can be increased.

4.2 Design Example

An example design has been carried out to illustrate the capabilities of this system and in particular to evaluate the embedded optimization techniques. A frigate of some 3,300 tonnes deep displacement (Δ_D) has been designed manually as a starting point for optimization, with the goal set as minimum resistance at design speed within the usual constraints (i.e., stability, strength, payload capacity, etc.). The initial hull-form, which is designed on a fixed displaced volume close to the light displacement, has a fairly high beam to draught ratio ($B_{WL}/T = 4.0$) and a low length to volume ratio ($\textcircled{m} = 7.0$). This allows the system scope to improve the design and also to avoid prejudging the 'best' solution. Of course, it is to be expected that rather predictable changes would be made to improve this design. However, this is a consequence of asking a question to which the answer is fairly well established. The real strength of optimizers lies in their ability to deal with many competing aspects at one time, whether or not the user is able to predict the likely consequences of attempting to satisfy such requirements, i.e., a known task has been specified to show that the optimizers can achieve the desired result without a priori knowledge. Also, for the sake of comparison, each different optimization method was tried from the same starting ship.

The initial input data for the frigate considered are given in Table 4.1 (with a summary of the corresponding output data in the first column of Table 4.5). The mission requirements for these designs are represented by a design speed of 30 knots in the deep condition, together with a payload consisting of a fuel deadweight of 600 tons at 0.8m above the keel plus a portion of the outfit weight which may be thought of as totalling 175 tons at a height which varies according to the hull and superstructure particulars. The fuel load is held fixed for simplicity, i.e., the maximum cruise range will vary; also the performance calculations are only to be carried out at full load displacement. Notice that the final deep displacement is not specified explicitly, since this is derived during the analysis of the hull, but instead the displaced volume has been given for the hull design (here, light) condition; these two differ substantially. This design meets all the criteria subsequently used as constraints during the optimization studies and the body plan is given in Figure 4.2.

To demonstrate the systems' abilities to deal with hull-form parameters the ship type was specified as FRIGATE, with $FLARE_X$, BT_{WL}/B_{WL} and L_X left as default values. ROF, L_{KEEL}, L_{MID} and $RAKE_A$ were given dimensional values rounded up from the equivalent non-dimensional default numbers and BT_{OA}/B_{OA}, $FLARE_A$, i_E, $i_{E\ UD}$ and $RAKE_F$ dimensional values which were then non-dimensionalised for the optimization process. This causes BT_{WL}/B_{WL}, $FLARE_X$ and L_X to take default non-dimensional values for the in-built frigate type and all the rest, new, non-dimensional values. The non-dimensional values then remain fixed so that their dimensional equivalents vary during optimization.

4.2.1 Selection of Objective Function, Constraints and Variables

The objective function, constraints and design variables selected for this problem are summarised in Table 4.2. As has already been mentioned, the objective function selected for minimisation was the total resistance at design speed. The constraints adopted were the six stability criteria (note that a non-standard, rather severe level has been specified for the initial GM criterion), an upper limit on the length to depth ratio (L_{WL}/D) to ensure against longitudinal strength problems and a minimum enclosed volume (∇_{ENC}) to guarantee payload capacity. Additionally L_{WL}, B_{WL}, T,D and C_x were constrained, but within very wide limits. The design variables chosen were those that would strongly affect both resistance and stability. These were (m), B_{WL}/T, C_P, $FLARE'_x$ (non-dimensional $FLARE_x$) and L'_x. Limits were chosen for these variables by making test runs, which established that values outside the given ranges would produce geometrically inconsistent ships. The system is able to cope with such inconsistencies, but these settings help it to run faster since it does not need to examine clearly unworkable combinations. In fact, some extreme combinations of variables, such as $C_P = 0.56$ and $L'_x = 0.3$, for (m) = 7.0, $B_{WL}/T = 4.0$ and $FLARE'_x = 0.0$, still produce inconsistent ships, and these cases were left in to exercise the system's ability to overcome such difficulties. Figure 4.3 illustrates the range of influence of L'_x on the hull-form, while Figure 4.4 shows that of extreme $FLARE_x$.

4.2.2 Two Dimensional Mapping of the Objective Function

In order to enable the objective function (R_T) to be mapped, according to the mapping strategy set out in Section 3.5, initially only (m) and B_{WL}/T were allowed to vary, the other variables remaining fixed as originally given, i.e., C_P = 0.57, L'_x = 0.52 and $FLARE'_x$ = 0. This mapping is shown in Figure 4.5 where it can be seen that resistance does not necessarily decrease continuously as the ship gets more slender, due to increase of the wetted area. When constrained by stability and structural strength, the compromise optimum lies in a region with B_{WL}/T a little greater than 3.4 and (m) at approximately 9.1, and this is entirely as expected for this reduced problem, being in line with the traditional practice of high speed frigate design. A sensitivity exercise is shown in Figure 4.6, where the ship is optimized at a different speed, here 15 knots. In this case, an unconstrained optimum is obtained at (m) $\approx$ 7.6 and $B_{WL}/T \approx 2.5$, indicating how speed sensitive such designs are. The detail mapping shows the previous constraint boundaries, which are unchanged with this different objective function. Here, only the stability constraint is active, indicating that a constrained optimum occurs at (m) $\approx$ 8.2 and $B_{WL}/T \approx 3.3$.

Still within the mapping strategy of Section 3.5, the other three variables of interest are combined to produce maps that demonstrate their influences. Figure 4.7, which is for (m) fixed at 7.0 and B_{WL}/T at 4.0 and Figure 4.8, for (m) at 7.0, show the influences of C_P, L'_x and $FLARE'_x$. These variables produce profound changes to the ship, and because of this can only be varied over rather limited ranges if

feasible designs are to result. It should be noted that the boundaries in Figure 4.7 are not explicit constraints, but rather the limits for geometrically consistent ships. In Figure 4.8, the left-hand boundary is caused by the application of the stability criteria, whilst the upper one is again the limit for consistent ships. The sloping left-hand boundary of Figure 4.8 indicates that increased flare increases stability allowing B_{WL}/T to be reduced and hence resistance decreased. Again, these trends might be as expected, but the best combination of all five variables is less apparent.

4.2.3 Optimization with Two Variables

Continuing with the previous strategy, the various optimizers are next applied to the two variable problem, varying (m) and $B_{WT.}/T$ with C_P, $FLARE'_x$ and L'_x fixed. The methods used were:

(1) Successive linear approximation - APPROX;

(2) Random exploration with shrinkage - RANDOM;

(3) Hooke and Jeeves direct search - SEEK,
with two types of penalty functions:

(a) one pass external function - OPTIM 1;

(b) Fiacco-McCormick combined external and internal function - OPTIM 2.

As mentioned in Chapter 3, each of the methods used has a number of control parameters to set up sizes, number of loops, tolerances, etc., all of which may be selected as required. With two variables, the maximum number of design loops was set at 1000 and all other features taken as default.

Given the previous contour maps of this problem the optimum is known and the purpose here is to establish that the optimizer could, in fact, reach this sub-goal. The results achieved are summarised in Table 4.3 and are next discussed in turn:

APPROX

This method gave a successful optimization, the minimum being reached very rapidly, in 24 loops (or ship designs). The optimization path is shown in Figure 4.9a and it can be seen that the expected minimum was reached (see Figure 4.5b).

RANDOM

With the maximum number of loops set to 1000, this routine fails to reach the expected minimum, although it indicated that it was satisfied with the result achieved. In fact, the optimizer stopped after 123 assessments of the objective function (OPTFUN, hull-form and resistance), which were preceded by 185 assessments of the constraints (OPTCON, hull-form and stability).

a) Using a one pass external penalty function (OPTIM 1) this method successfully reached the optimum. When constraint lines were violated (at first stability and subsequently L_{WL}/D), the penalty function successfully drove the variables back to the feasible area. After 90 loops, 100 random points were tried in the vicinity of the last result, and the best result used as a new starting point. After 369 loops, including a second set of 100 random designs, it finally reached the minimum. These results are very similar to those generated by APPROX. The path followed, including the shotgun searches, can be seen in Figure 4.9b.

b) Using the Fiacco-McCormick combined external and internal function (OPTIM 2) this method again gave a successful optimization, using the same strategy as with OPTIM 1, but with a smoother penalty function. When using this penalty function there is no random search, as it is designed to avoid working far from the feasible region. The process ended after 613 loops, requiring more steps because of the repeated sub-optimizations used. The results are very close to those given by SEEK with OPTIM 1 and also APPROX. The general path of the optimization process is shown in Figure 4.9c.

The only problem encountered with these two dimensional optimizations concerns the failure of RANDOM to reach the true optimum, especially since this routine ought to provide a standard for comparison. This failure can be directly attributed to the shrinkage mechanism, combined with the small number of points tested between each reduction in the search area. This mechanism causes the random search to be concentrated in areas found to be giving good results, allowing the true optimum to be rejected under some circumstances. This arises because infeasible points are rejected, irrespective of the objective-function value at the location tested; consequently this method tends to miss optima that are defined by constraint boundaries, as here. Nonetheless, it should be noted that the routine still produced a reduction in resistance of some 30% compared to the original design. To overcome this difficulty, and to increase confidence in the method, the shrinkage mechanism must be made less severe. By default, it takes ten times as many points as variables at each stage, of which it keeps the best 25% (i.e., five from twenty for this two dimensional problem) and this set of best points is used to construct the next shrunken range for investigation. Increasing the total sample to eighty whilst still retaining the best five, yields the desired result, although the exact precision of the other methods is still not achieved, see again Table 4.3.

4.2.4 Optimization with Five Variables

Having established under what conditions the optimizers work with two variables and also the likely effects of modifying the different variables under investigation, it is possible to move on to the full problem, i.e., a five parameter optimization. Although this multidimensional process cannot be mapped, it is to be expected that the best ship would lie in a region where (m) is about the same as for the two dimensional problem. B_{WL}/T might be slightly smaller here combined with increased flare to restore stability. Also a minimum prismatic coefficient is probable, combined with a position of maximum beam set well aft (this shifts the longitudinal centre of buoyancy aft and also tends to produce a smaller angle of entrance for the water-plane). These last three variables produce profound changes in the hull-form, compared to the simple stretching caused by altering (m) and B_{WL}/T. Therefore, the optimization becomes heavily constrained because of the large number of geometrically inconsistent hull-forms that can arise. This aspect tends to produce a narrow corridor of feasible designs through the multidimensional space, which may be hard for the optimizers to enter or stay within. Clearly the five dimensional problem is considerably more taxing for the method; the results achieved by the various optimizers are given in Table 4.4 and are again discussed in turn:

APPROX

This method fails after only six loops through the design process. It is completely unable to cope with geometrically impossible ships for which correct information cannot be given concerning the objective function. As has already been mentioned, the system attempts to overcome such impossible combinations of parameters by synthesising values for the objective function and constraints. When only one impossible design is encountered whilst linearising the objective function, this just produces a massive distortion directing the search away from the trouble-spot; this happened occasionally during the two dimensional search. However, when several such combinations occur, as happens with the five dimensional search, the resulting hyper-plane no longer points back towards sensible designs, and moreover loses all similarity to a smooth surface when the next linearised region is being constructed. The search gets hopelessly confused and gives up. It is difficult to see a way around this dilemma, this being a fundamental shortcoming of this otherwise powerful technique when applied to highly constrained problems.

RANDOM

RANDOM suffers from no such drawbacks, since it is unconcerned with the relationships between the various points tested, merely keeping a best subset. This method was applied allowing up to 5,000 combinations of the five variables, since it was found that it needed to try a very large number of combinations to get near to the minimum, many

combinations being rejected due to infeasibility. As can be seen from Table 4.4, the search did not find the true minimum resistance with the default shrinkage mechanism. As has already been noted, if the random search does not hit a point in the vicinity of the best results at least once, the shrinkage technique used to improve speed tends to throw away the whole region, concentrating the search in the area where the best feasible point was found. The general reliability of a pure random search is thus jeopardised when the shrinkage technique is introduced to make it work more quickly. Consequently, when the function being optimized is such that one cannot predict in which region of the n-dimensional space the optimum lies, the shrinkage technique should be reduced in severity. The resulting runs therefore tend to be extremely long, but the optimum can always be found. By default, with five variables RANDOM takes fifty samples and keeps the best thirteen. Increasing the total to 200 and keeping the best sixteen causes the method to examine many more combinations, but in the end it does achieve the desired result (see Table 4.4).

SEEK

a) Using a one pass external penalty function (OPTIM 1) this method succeeded after 539 designs, the direction of search being decided, as before, by trying changes to the variables, one at a time. This approach produces a consistent path leading towards the optimum. When inconsistent hull-forms were encountered the modified objective function already discussed drove the search successfully back on course.

b) Using the Fiacco-McCormick combined external and internal penalty function (OPTIM 2) the method failed to find a satisfactory optimum with five variables. However, the method did produce the design with least resistance, but with failure indicated due to the violated limit of the C_p component of the trial vector. The fact that this violation is not great indicates that with further interactions the process would probably converge, although this is not carried through here.

As predicted, the various optimization processes led to approximately the same optimum design, according to the restrictions imposed by the various constraints and the choice of variables. The process tends to produce a longer and more slender ship increasing the length and decreasing the beam up to a certain extent. The limit reached is very much one of compromise between reducing wave making resistance and loss of strength and/or stability. The most rapid optimization, within the specified constraints, was achieved for five variables by the Hooke and Jeeves direct search with a one-pass-external penalty function. The final design is illustrated in Figure 4.10. The improvement in resistance, from the original starting point, is of the order of 81% when varying (m) and B_{WL}/T only, with a further decrease of 3% being achieved when C_p, the position of maximum beam and flare were all varied. Note also that this is achieved despite the increase in deep displacement of some 200 tonnes caused by the changes in hull dimensions. Again, it must be emphasised that these rather predictable results arise because a well known problem has been examined, the aim

being to demonstrate the technique in a realistic setting. The details of the final designs achieved using all the methods described here are given in Table 4.5, along with those for the original design.

4.2.5 Analysis

The use of the optimization process in this design example and other exercises confirmed how time consuming optimization can be, that a user unacquainted to optimization and these methods would have difficulties in setting the objective function and constraints properly and that it is very difficult to judge what action to take when the optimization fails. Even if it is successful, there is uncertainty as to whether the results produced are accurate. However, given the behaviour of the methods, some improvement in shortening the process and for helping with subsequent decision-making can be suggested.

As was expected, the different methods gave approximately similar results and the paths followed were also generally similar, as can be seen from the two dimensional mappings (and also by analysing the step pattern in the output files of the five dimensional problem). This is not at all surprising since the methods are starting from the same point. This fact suggests that, after a failure, when starting another try or when changing methods it is worthwhile using the last best design as the new starting point. The new process will start with an initial step size of increased size and this could help it to get out of the trouble that stopped the previous run. Features such as step size and number of runs, etc. can also be modified.

It was also noticed that some runs suggest improvements if certain constraints are relaxed. When setting the problem it may not be clear that some constraints are overly restrictive and this becomes obvious only when certain unexpected favourable combinations of parameters arise in the search process.

In terms of time consumption, Siddall's view was confirmed, that methods like APPROX which are extremely fast are also more liable to failure than others, while the most reliable ones, like RANDOM take the longest. This suggests that the quickest methods should be tried until exhausting their possibilities, before moving on to the others.

The general rule-of-thumb seems to be that, for best efficiency, the fastest methods should be tried first; then, as they fail, the last best designs should be taken as new starting points. Also these faster methods should be tried until all their possibilities are exhausted, before moving on to slower techniques working over reduced ranges.

Heuristic methods such as the Hooke and Jeeves direct search are very efficient methods for design applications[30] because their way of searching is similar to that found successful in normal design reasoning and so they should be pursued. However, the linear approximation method should be tried first due to its great speed. The mapping strategy is also very useful in helping the user understand the optimization process. Nonetheless, some further guidance is necessary on what

hierarchy of decision-making would be most efficient, such as what to run first, what features to modify and how, searches for false optima, where to make local random searches, when to relax constraints, etc.

Advice on such topics would seem to lie in the domain of artificial intelligence and so such techniques are considered in the remaining chapters.

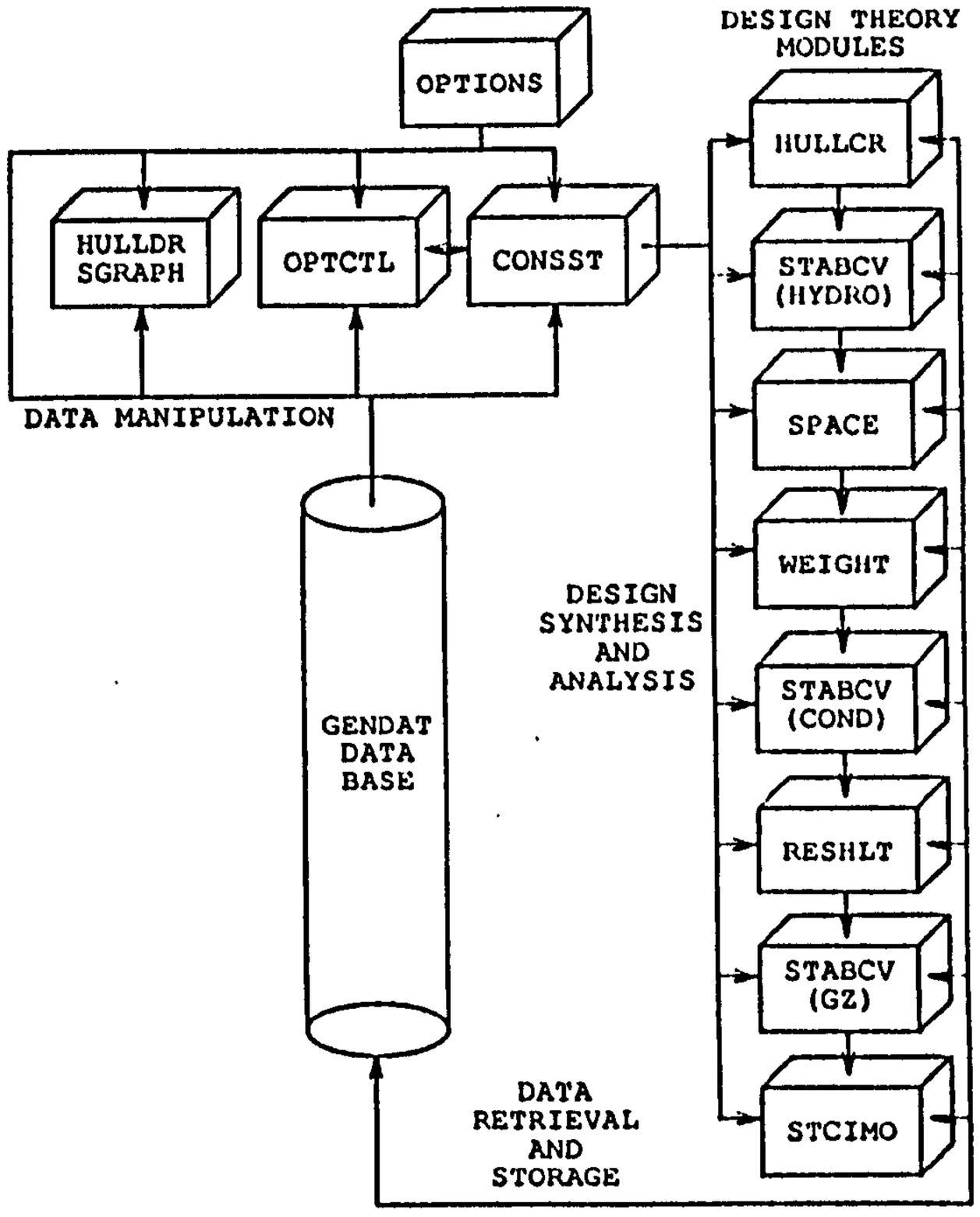

Figure 4.1 - Design system structure.

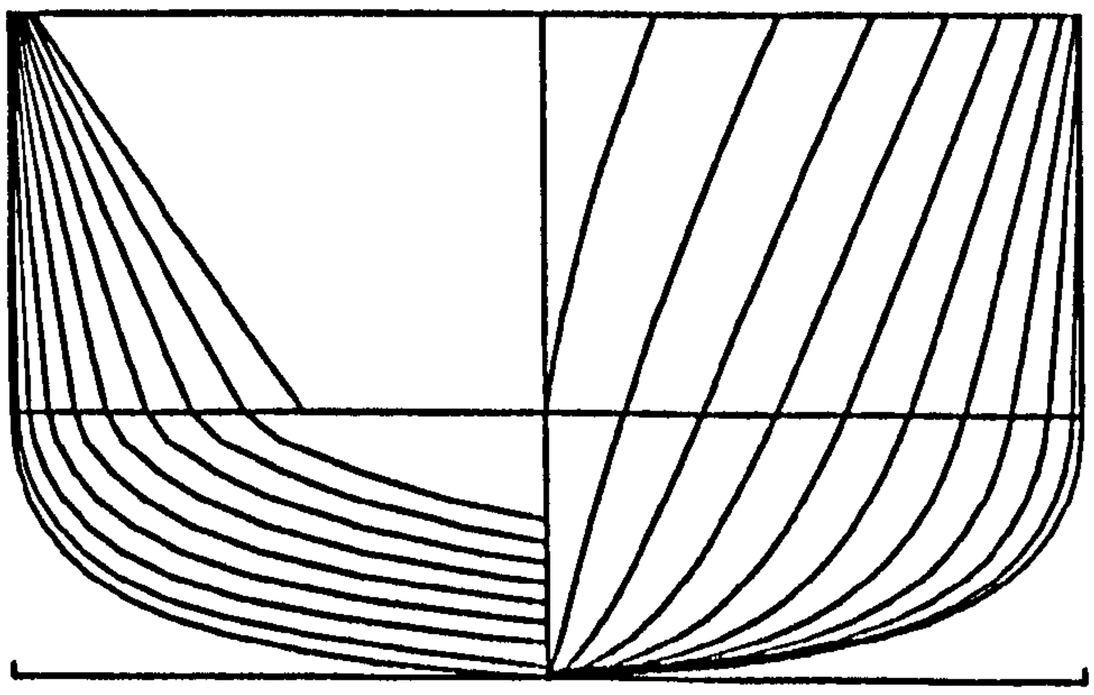

Figure 4.2 - Body plan for the initial design.

ⓜ	7.0	C_P	0.57
B_{WL}/T	4.0	∇	2400 m^3
C_B	0.47	D/T	2.5

Table 4.1 - Initial input data.

Optimization	with 2 variables	with 5 variables
Objective Function	Total Resistance R_T (KN)	
Constraint Vector and Limits	Initial GM $\geq$ 0.7 m. Area below GZ curve from 0 to 40° $\geq$ 0.09 m rad. Area below GZ curve from 30 to 40° $\geq$ 0.03 m rad. Area below GZ curve from 0 to 30° $\geq$ 0.055 m rad. Angle of Maximum GZ $\geq$ 30°. Maximum GZ $\geq$ 0.2 m. $L_{WL}/D \leq 14.0$ $12000m^3 \leq \nabla_{ENC}$ $0.0 \leq L_{WL} \leq 1000.0$ m $0.0 \leq B_{WL} \leq 1000.0$ m $0.0 \leq T \leq 1000.0$ m $0.0 \leq D \leq 1000.0$ m $0.4 \leq C_X \leq 1.0$	
Trial Vector and Limits	$5.0 \leq$ ⓜ ≤ 12.0 $1.0 \leq B_{WL}/T \leq 6.0$ $C_P = 0.57$ $FLARE'_X = 0.0$ $L'_X = 0.52$	$5.0 \leq$ ⓜ ≤ 12.0 $1.0 \leq B_{WL}/T \leq 6.0$ $0.55 \leq C_P \leq 0.60$ $0.0 \leq FLARE'_X \leq 10.5$ $0.14 \leq L'_X \leq 0.60$
Fixed Parameters	$\nabla = 2400m^3$ $C_B = 0.47$ $D/T = 2.5$	

Table 4.2 - Optimization Conditions (Summary).

Figure 4.3a - Variation of non-dimensional position of maximum beam, L'_X (aft).

Figure 4.3b - Variation of non-dimensional position of maximum beam, L'_X (forward).

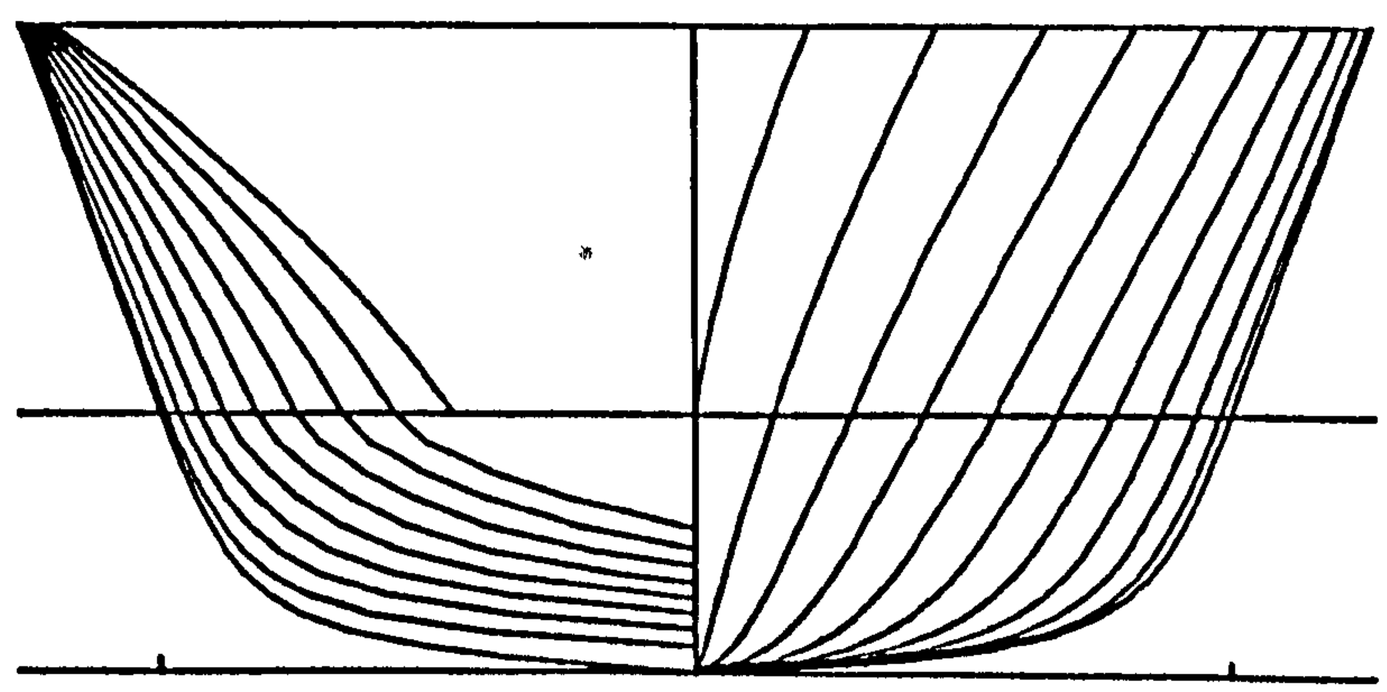

Figure 4.4- Variation of waterline flare at max. beam, $FLARE_X$.

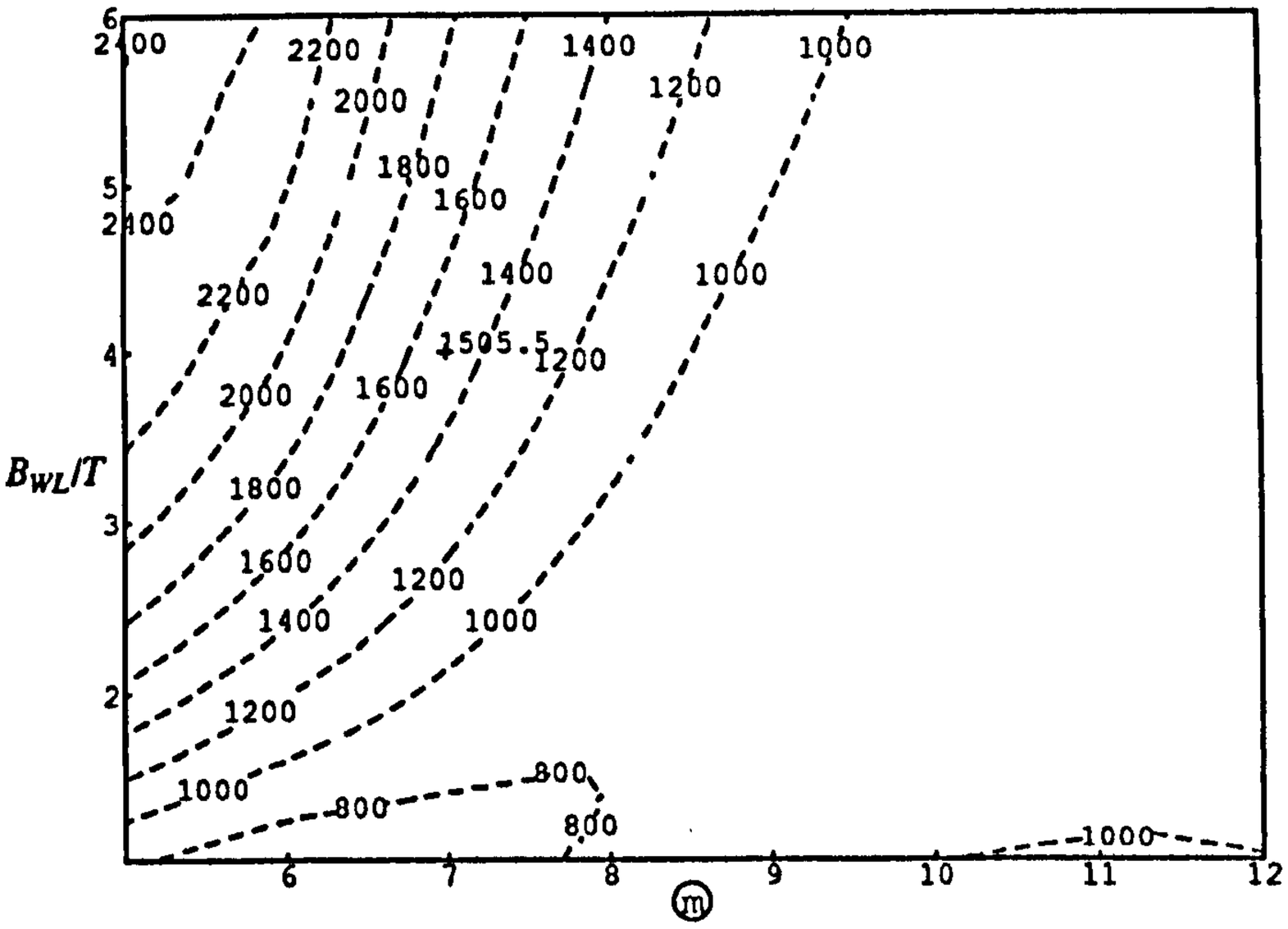

Figure 4.5a - Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, (m) versus waterline beam to draught ratio, B_{WL}/T.

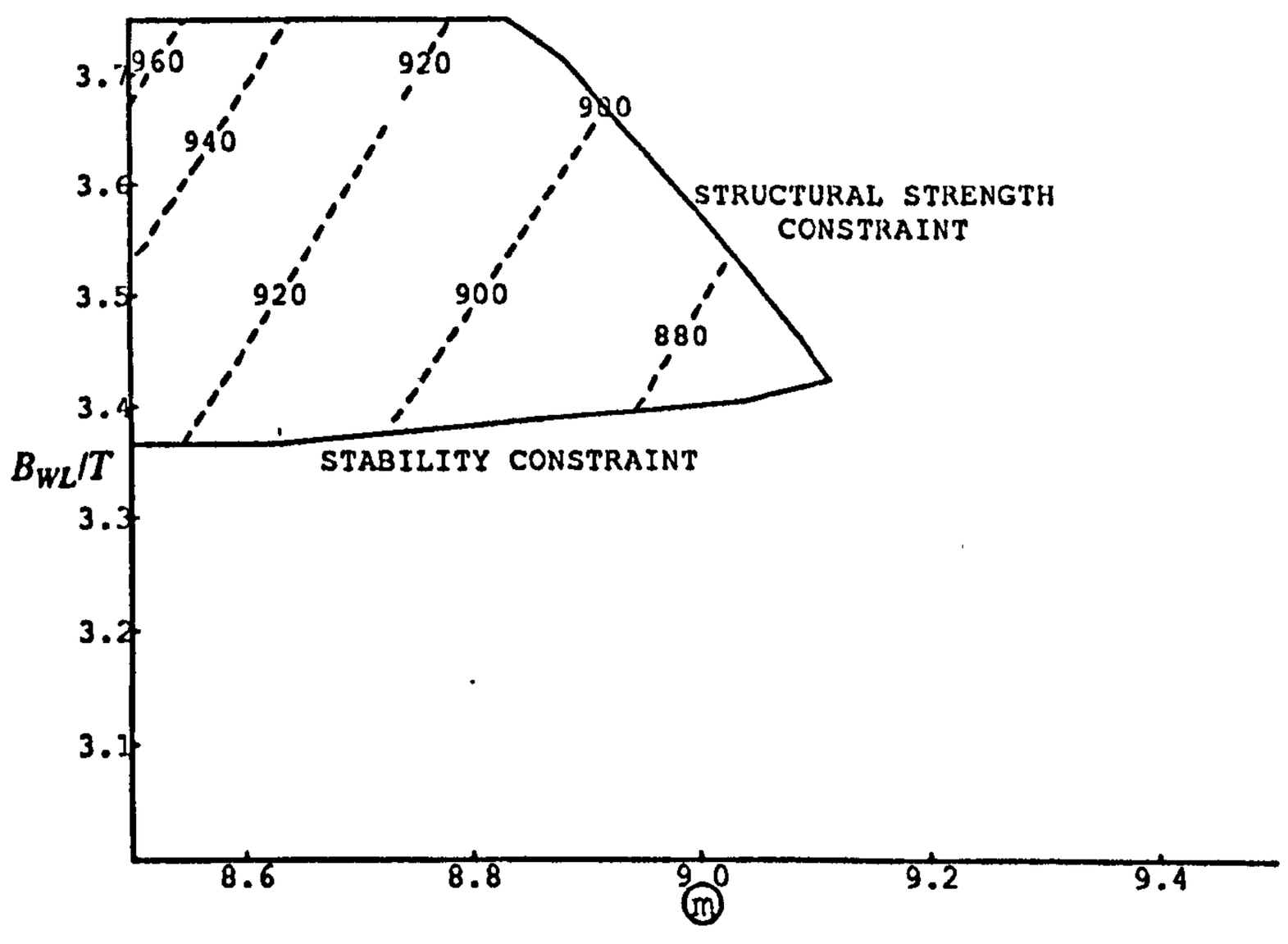

Figure 4.5b - Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, (m) versus waterline beam to draught ratio, B_{WL}/T (taken from Figure 4.5a but with constraint boundaries shown).

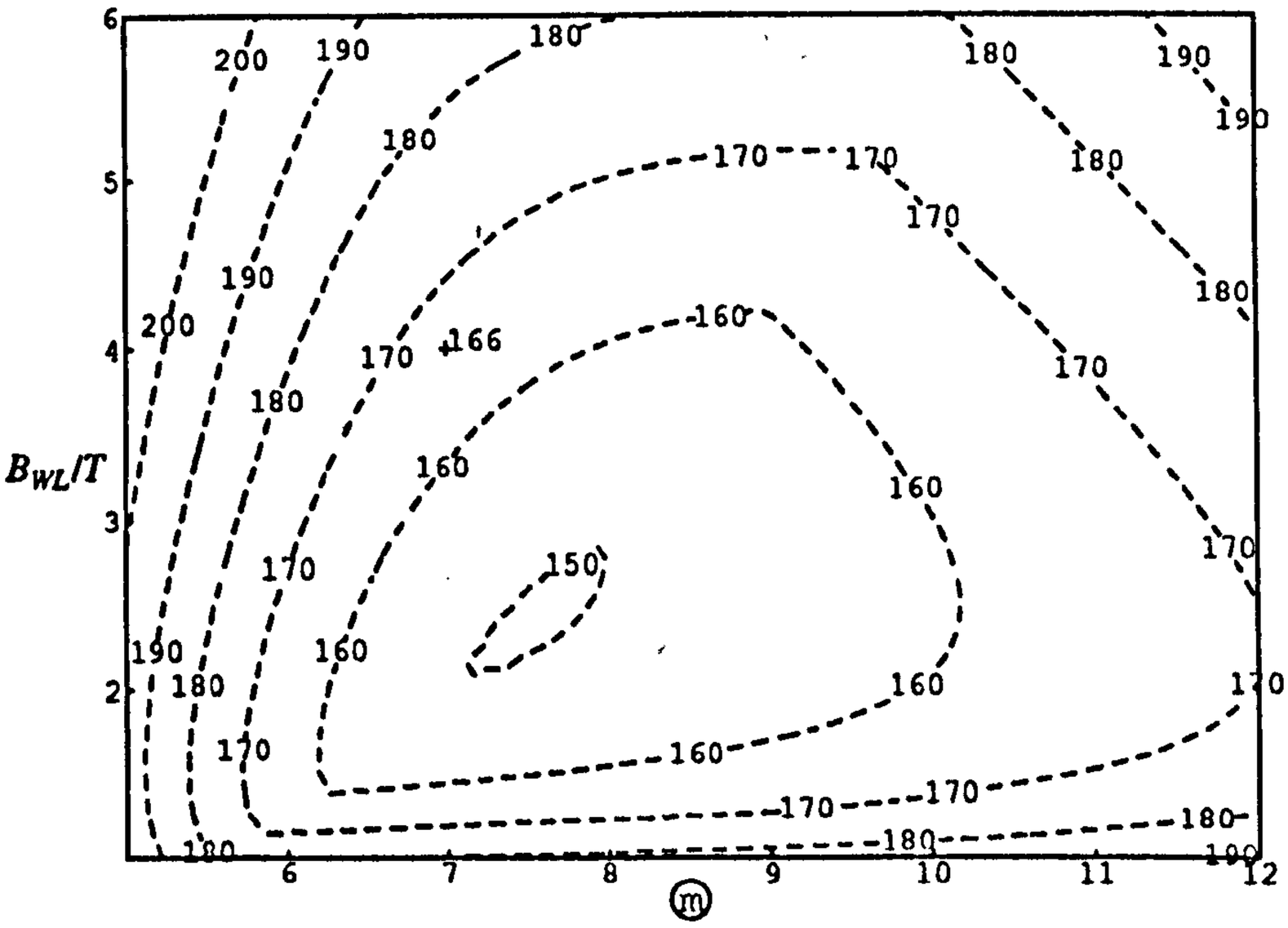

Figure 4.6a - Contour map of total resistance, R_T (kN) at 15 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T.

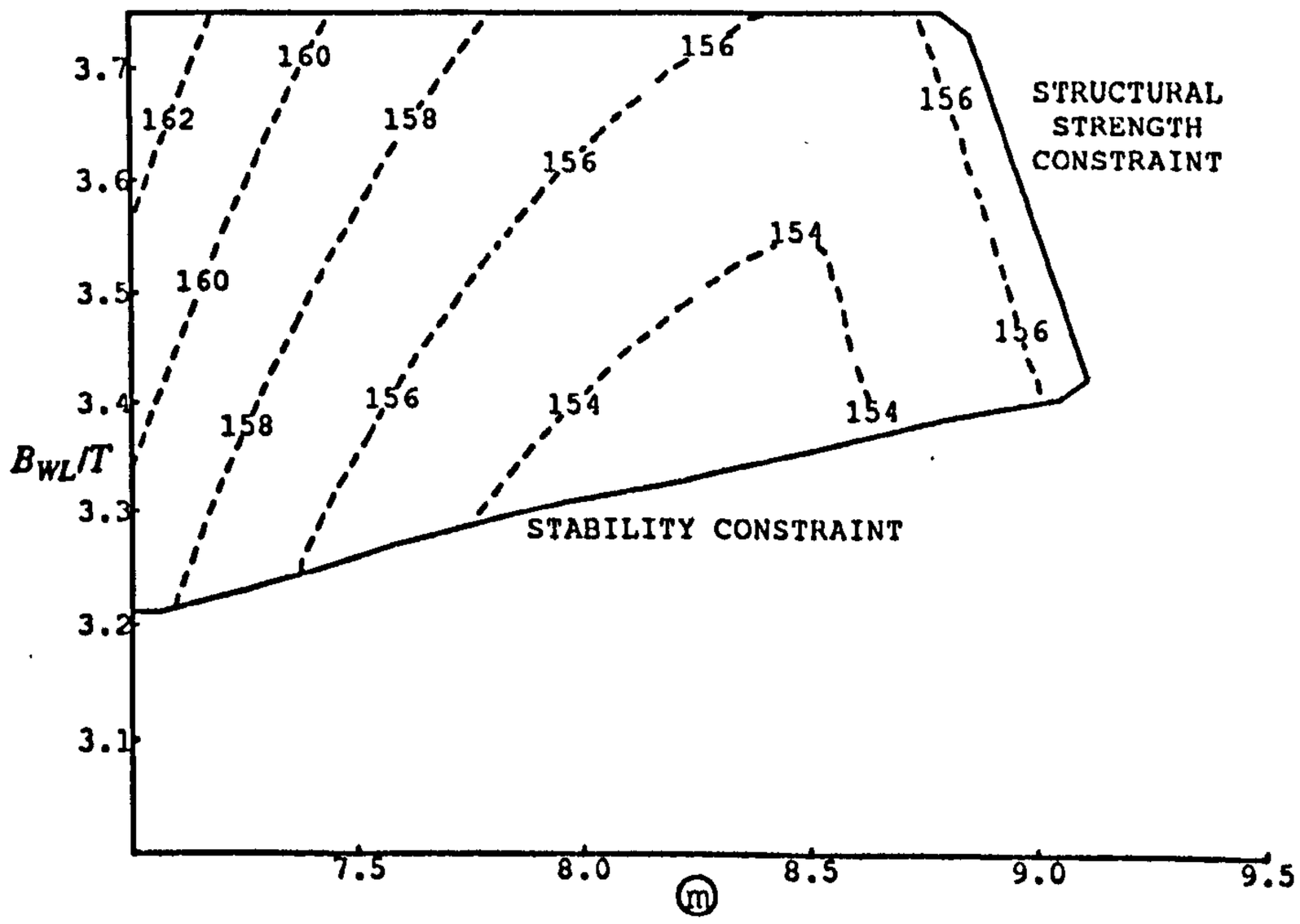

Figure 4.6b - Detail contour map of total resistance, R_T (kN) at 15 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T (taken from Figure 4.6a but with constraint boundaries shown).

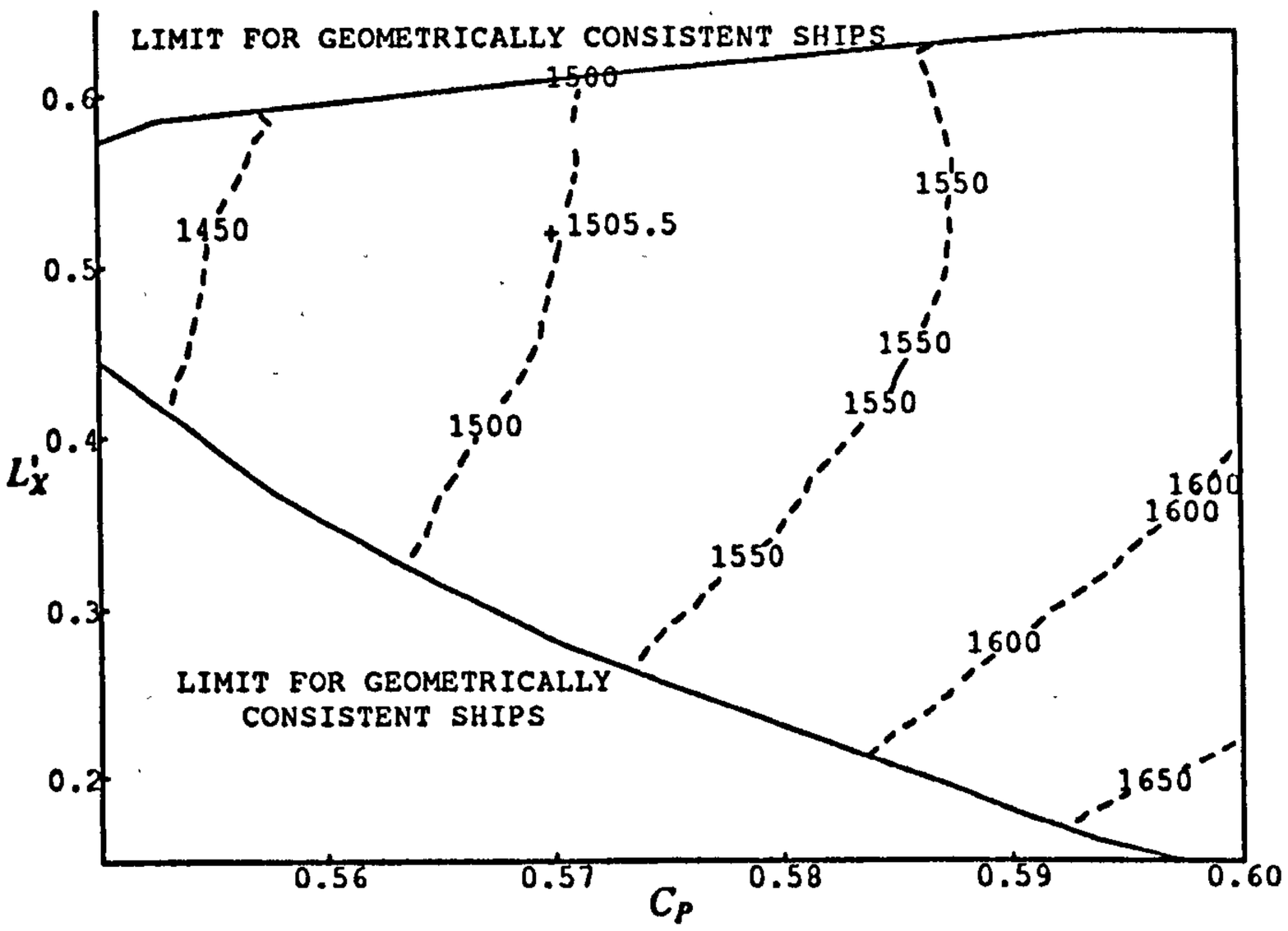

Figure 4.7 - Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L'_X.

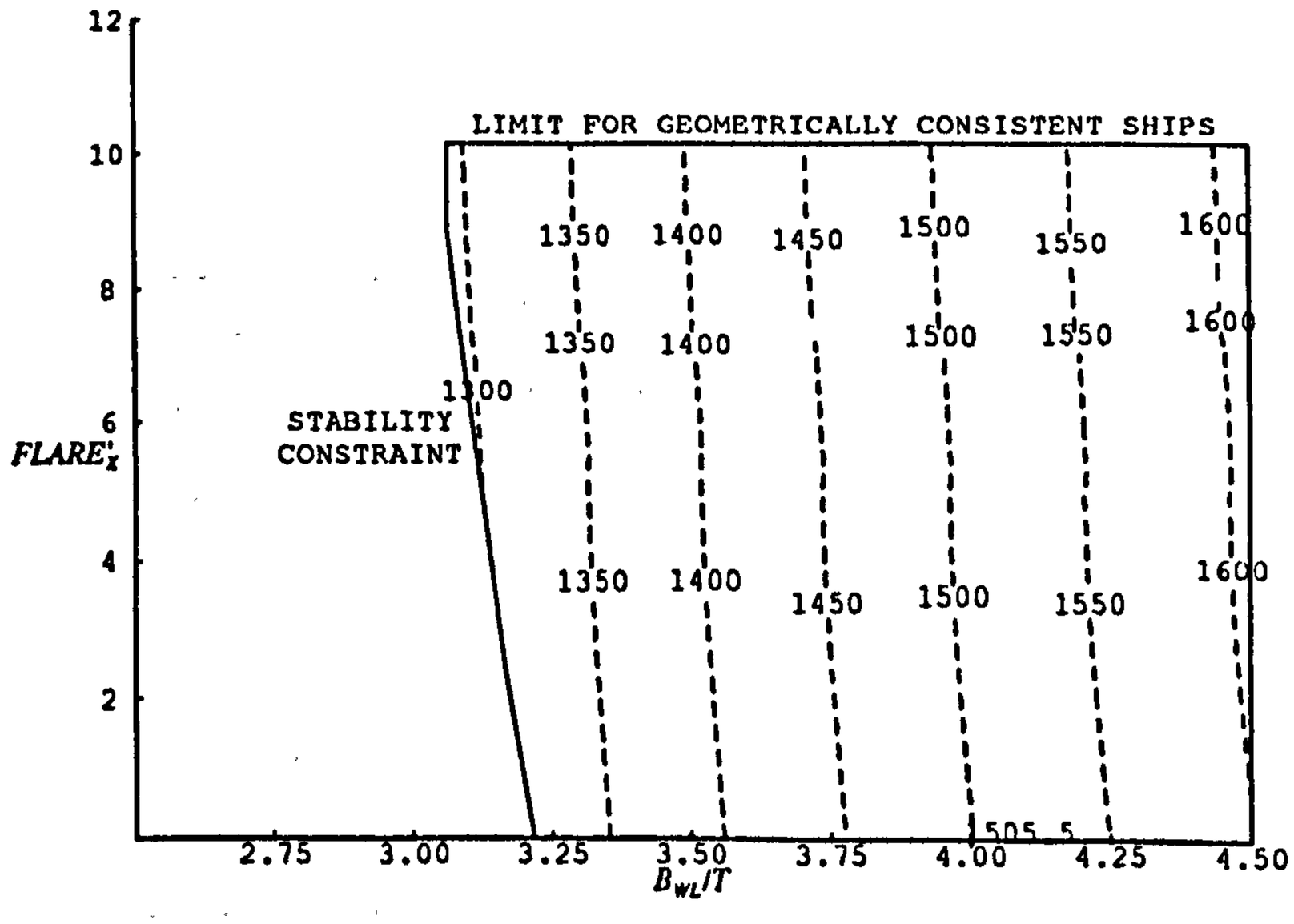

Figure 4.8 - Contour map of total resistance, R_T (kN) at 30 kts. for waterline flare at maximum beam, $FLARE'_X$ versus waterline beam to draught ratio, B_{WL}/T.

Method		APPROX	RANDOM	SEEK	SEEK	RANDOM *
Penalty Function		n/a	n/a	OPTIM1	OPTIM2	n/a
Success/Failure		Success	Success	Success	Success	Success
Number of Loops		24	185 †	369	613	503 †
Objective Function	R_T (KN)	855.0	1064.9	855.1	855.1	876.8
Trial Vector	Ⓜ	9.143	7.974	9.143	9.143	8.906
	B_{WL}/T	3.410	3.542	3.410	3.411	3.399
Constraint Vector	Initial GM (m)	0.70	1.06	0.70	0.70	0.71
	Area 0-40° (m rad)	0.23	0.31	0.23	0.23	0.23
	Area 30-40° (m rad)	0.11	0.14	0.11	0.11	0.11
	Area 0-30° (m rad)	0.12	0.16	0.12	0.12	0.12
	Angle Max GZ (°)	54	53	54	54	54
	Max GZ (m)	0.92	1.14	0.92	0.92	0.92
	L_{WL}/D	14.0	11.6	14.0	14.0	13.4
	$\nabla_{ENC}(m^3)$	13588	13698	13588	13588	13582

Table 4.3 - Optimization with Two Variables.
* modified shrinkage action, see text. † objective function evaluated at feasible combinations only.

Method		APPROX	RANDOM	SEEK	SEEK	RANDOM *
Penalty Function		n/a	n/a	OPTIM1	OPTIM2	n/a
Success/Failure		Failure	Success	Success	Failure	Success
Number of Loops		6	1395 †	539	1926	6336 †
Objective Function	R_T (KN)	-	923.6	832.0	829.2 $	835.0
Trial Vector	Ⓜ		8.370	9.149	9.148	9.220
	B_{WL}/T		3.314	3.396	3.404	3.303
	C_P	-	0.562	0.550	0.547	0.558
	$FLARE'_X$		1.5	0.4	0.0	4.7
	L'_X		0.483	0.581	0.580	0.566
Constraint Vector	Initial GM (m)		0.72	0.70	0.70	0.70
	Area 0-40° (m rad)		0.24	0.23	0.23	0.25
	Area 30-40° (m rad)		0.12	0.12	0.12	0.13
	Area 0-30° (m rad)	-	0.12	0.12	0.12	0.12
	Angle Max GZ (°)		55	54	53	54
	Max GZ (m)		1.04	0.97	0.96	0.98
	L_{WL}/D		12.1	14.0	14.0	14.0
	$\nabla_{ENC}(m^3)$		13820	13647	13580	14425

Table 4.4 - Optimization with Five Variables.
* modified shrinkage action, see text. † objective function evaluated at feasible combinations only.
$ last result with sufficient stability before failure.

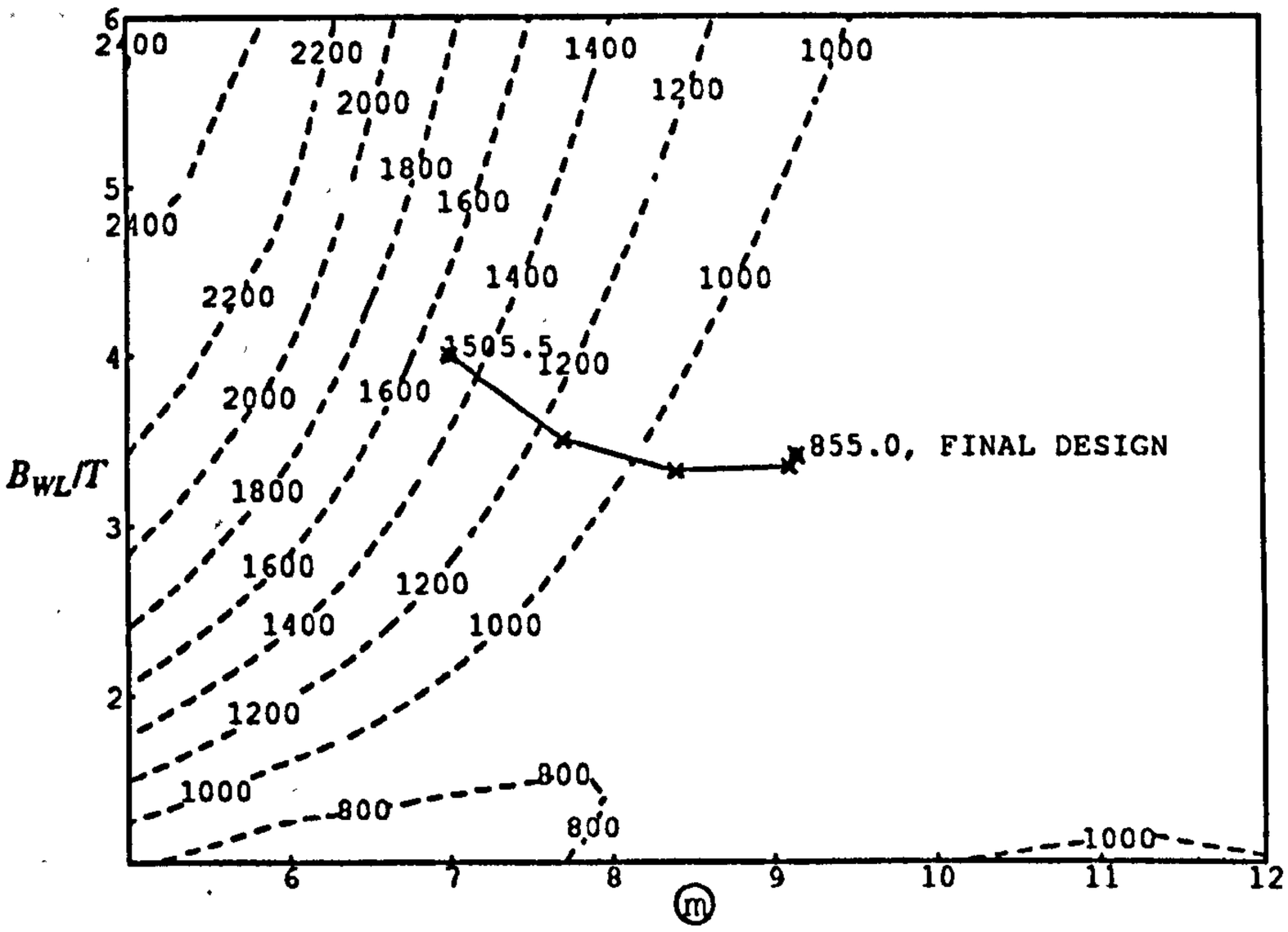

Figure 4.9a - Optimization path for APPROX with two variables.

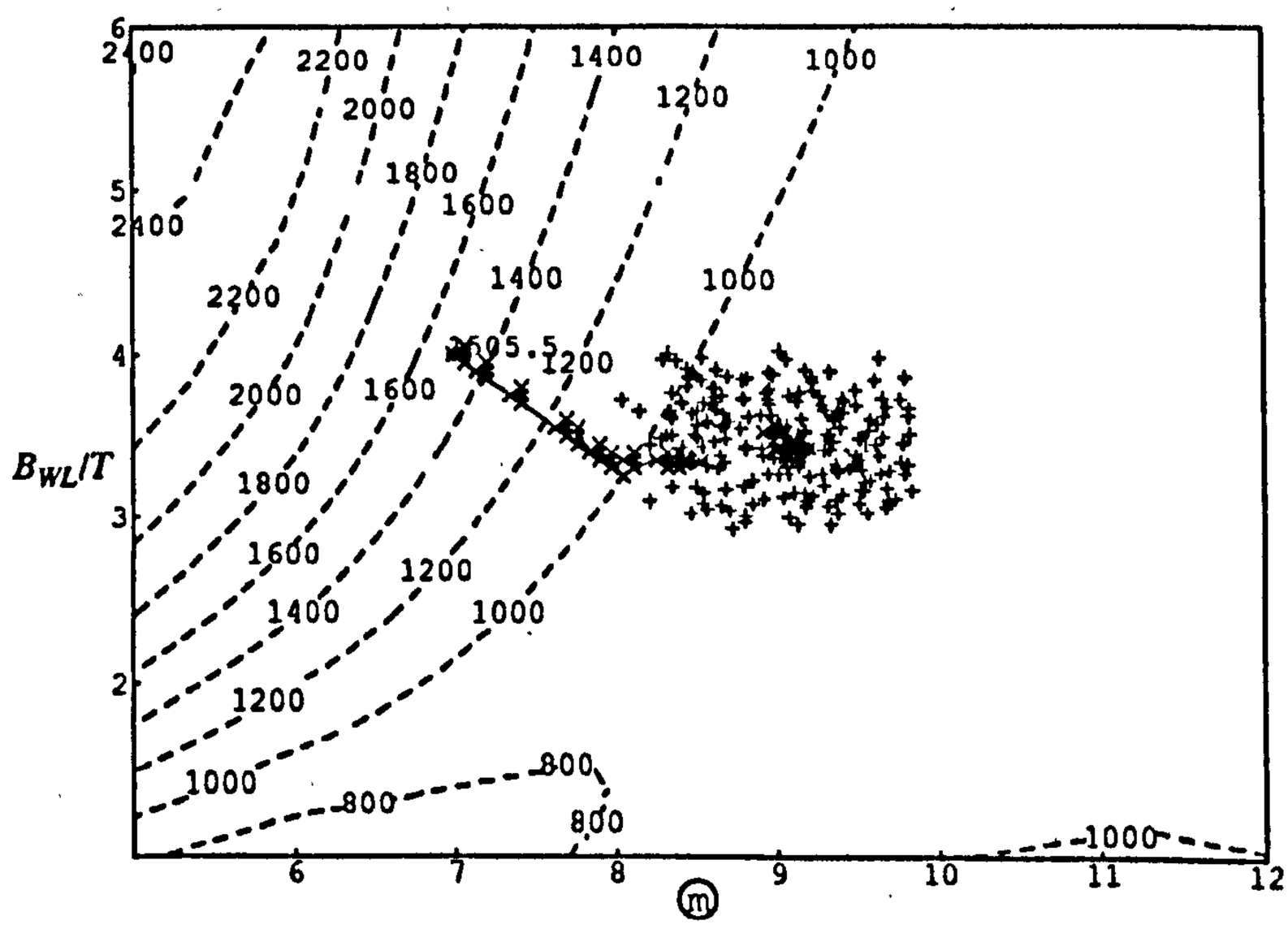

Figure 4.9b - Optimization path for SEEK and OPTIM1 with two variables.
(Points marked '+' are produced by the shotgun searches - see appendix A.)

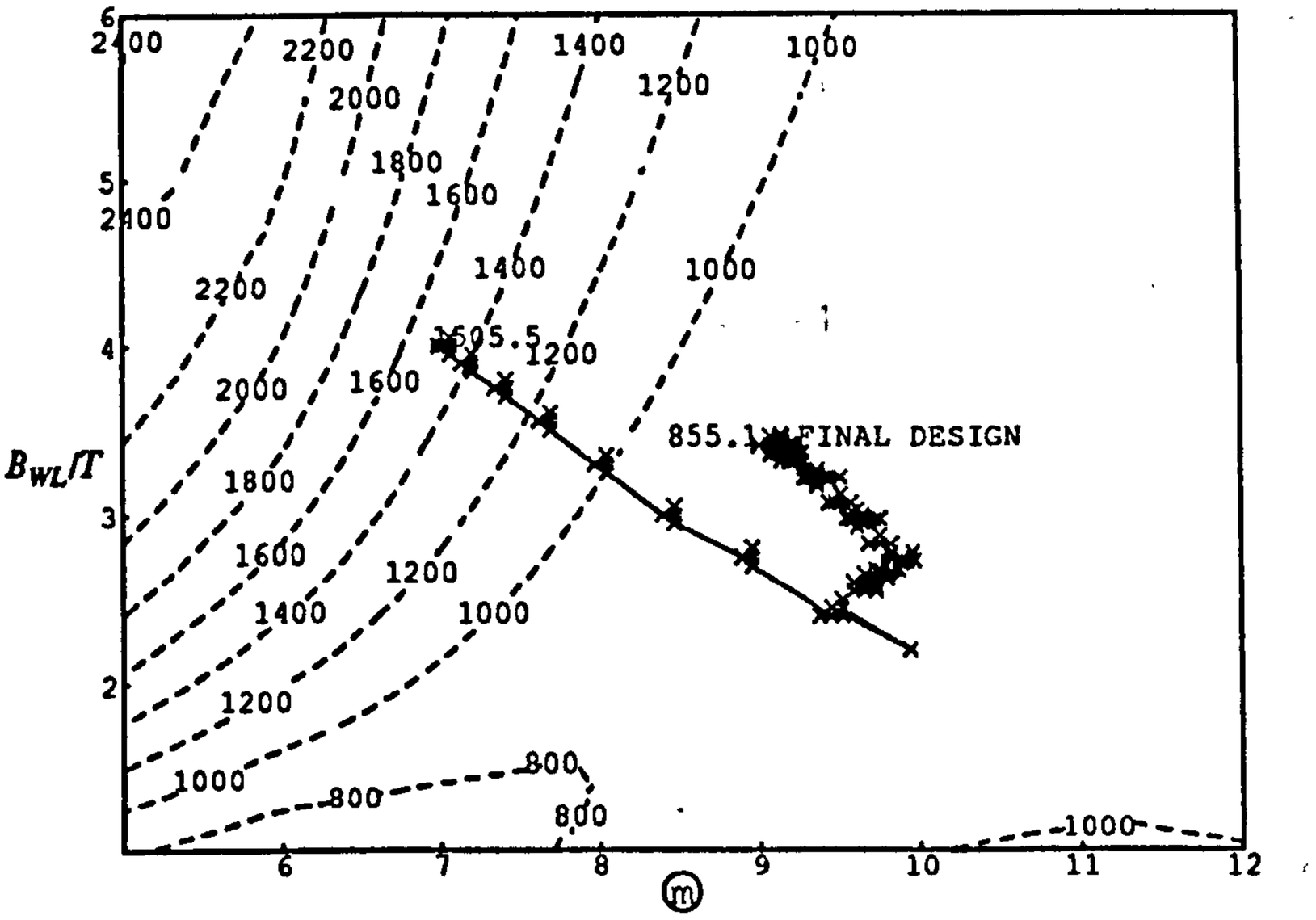

Figure 4.9c - Optimization path for SEEK and OPTIM2 with two variables.

RESULTS	Startship	Optimization with 2 variables				Opt. with 5 variables	
Method		APPROX	RANDOM *	SEEK	SEEK	RANDOM *	SEEK
Penalty Function		n/a	n/a	OPTIM1	OPTIM2	n/a	OPTIM1
R_T (KN)	1505.5	855.0	876.3	855.1	855.1	835.0	832.0
ⓜ	7.0	9.143	8.906	9.143	9.143	9.220	9.149
B_{WL}/T	4.0	3.410	3.399	3.410	3.411	3.303	3.396
C_P	0.57	0.57	0.57	0.57	0.57	0.558	0.550
$FLARE'_X$	0.0	0.0	0.0	0.0	0.0	4.7	0.4
L'_X	0.52	0.52	0.52	0.52	0.52	0.566	0.581
L_{WL} (m)	93.8	122.5	119.3	122.5	122.5	123.5	122.6
B_{WL} (m)	14.77	11.93	12.07	11.93	11.93	11.69	11.91
T (m)	3.693	3.499	3.551	3.499	3.499	3.541	3.505
D (m)	9.23	8.75	8.88	8.75	8.75	8.85	8.76
C_X	0.825	0.824	0.825	0.824	0.824	0.842	0.855
C_{WP}	0.722	0.722	0.722	0.722	0.722	0.720	0.719
B_{OA} (m)	14.77	11.93	12.07	11.93	11.93	13.15	12.02
BT_{WL}/B_{WL}	0.45	0.45	0.45	0.45	0.45	0.45	0.45
BT_{OA}/B_{OA}	0.95	0.95	0.95	0.95	0.95	0.95	0.95
$FLARE_X$ (°)	0.0	0.0	0.0	0.0	0.0	7.8	0.6
$FLARE_A$ (°)	35.0	30.8	30.8	30.8	30.8	30.0	30.7
ROF (°)	2.5	2.9	2.9	2.9	2.9	3.0	2.9
L_{MID} (m)	1.80	2.35	2.29	2.35	2.35	2.37	2.35
L_{KEEL} (m)	52.0	67.9	66.2	67.9	67.9	68.5	68.0
i_E (°)	13.0	8.1	8.4	8.1	8.1	7.9	8.1
$i_{E\,UD}$ (°)	25.0	16.0	16.7	16.0	16.0	15.7	16.0
$RAKE_F$ (°)	33.0	41.8	40.7	41.8	41.8	41.7	41.8
$RAKE_A$ (°)	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Initial GM (m)	2.01	0.70	0.71	0.70	0.70	0.70	0.70
Area 0-40° (m rad)	0.52	0.23	0.23	0.23	0.23	0.25	0.23
Area 30-40° (m rad)	0.23	0.11	0.11	0.11	0.11	0.13	0.12
Area 0-30° (m rad)	0.29	0.12	0.12	0.12	0.12	0.12	0.12
Angle Max GZ (°)	49	54	54	54	54	54	53
Max GZ (m)	1.57	0.92	0.92	0.92	0.92	0.98	0.97
L_{WL}/D	10.2	14.0	13.4	14.0	14.0	14.0	14.0
$\nabla_{ENC}(m^3)$	14065	13588	13605	13588	13588	14425	13647
Δ_D (t)	3274	3492	3462	3492	3492	3490	3491

Table 4.5 - Details of the Final Designs.
* modified shrinkage action, see text.

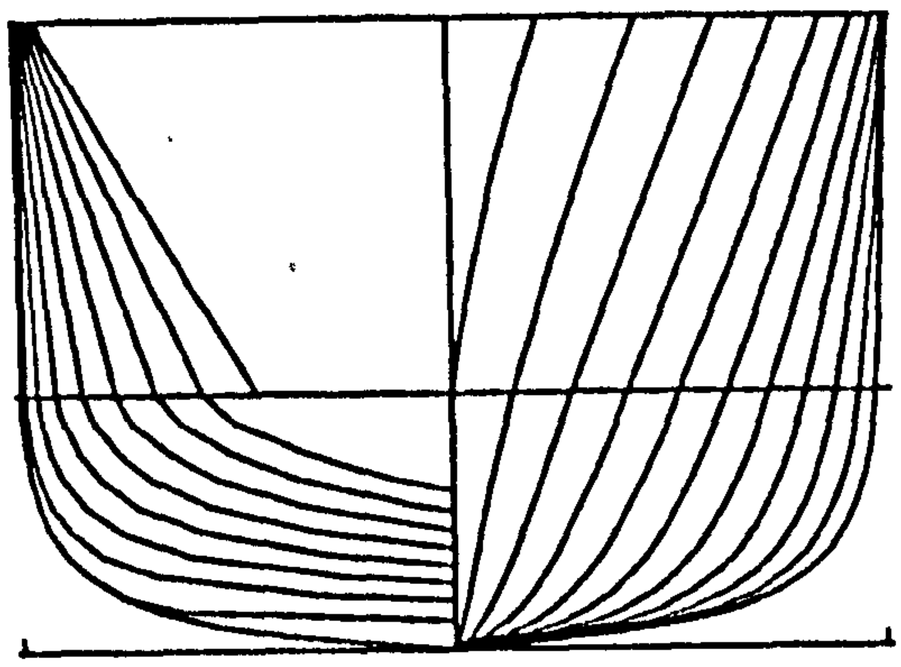

Figure 4.10a - Body plan for the best, constrained design.

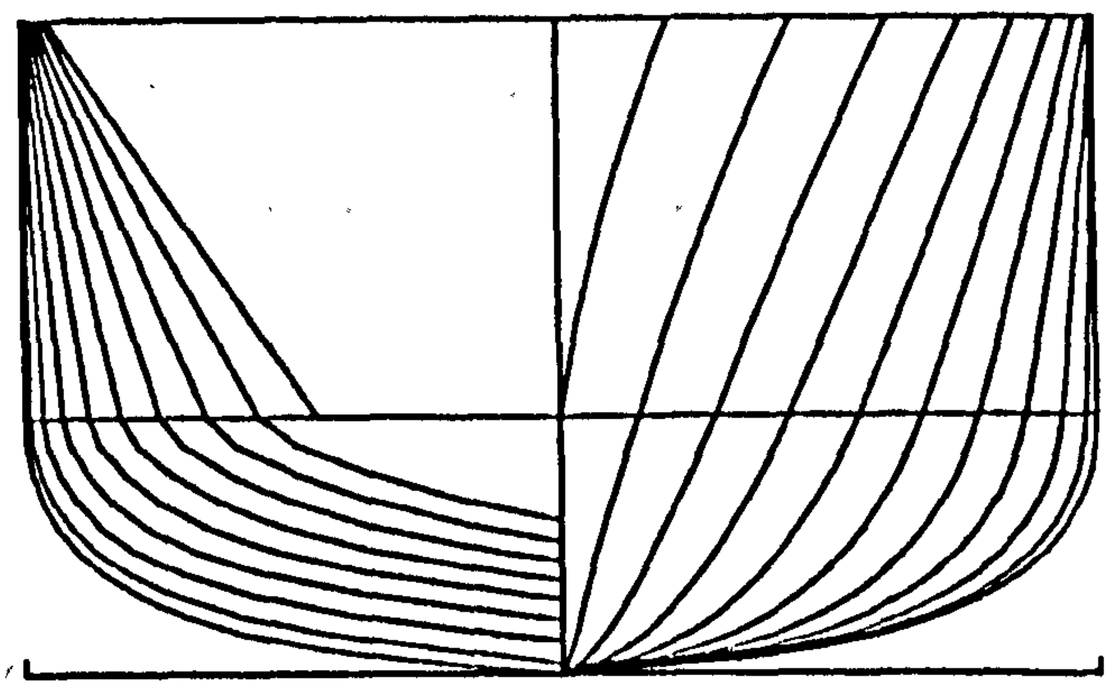

Figure 4.10b - Body plan for the initial design.

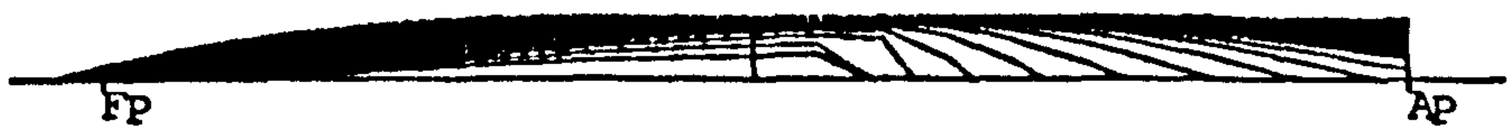

Figure 4.10c - Water-planes for the best, constrained design.

Figure 4.10d - Water-planes for the initial design.

5. KNOWLEDGE BASED SYSTEMS

5.1 Introduction

As noted in Chapter 3, it would seem that a Knowledge Base containing expert advice to control the optimization process using a mapping strategy would make possible the best use of these techniques when applied to the Design Process. Therefore, although Expert Systems have found wide applications in a number of different areas, including design, the application of such a technique, is here restricted to the control of the optimization process. Besides such control of optimization, a few other possible applications of Expert Systems will also be briefly studied, in order to develop an idea of their applicability and potential for future integration into the ship concept design process. Before entering into such discussions a brief review of Artificial Intelligence and Expert Systems will be given.

5.2 Definitions

Artificial Intelligence and Expert Systems cover a vast area of study, principally in the realm of Computer Science. In this work only a few definitions from the area will be given, being just enough to introduce the subject and to help illustrate its application.

5.2.1 Knowledge Based Systems

According to Jackson[40] knowledge based systems are systems which solve problems by applying a symbolic representation of human expertise, instead of employing more algorithmic or statistical methods. In other words, knowledge based systems attempt to encode the domain-specific knowledge of every-day practitioners in some field, rather than using complex and comparatively domain-free methods derived from science or mathematics.

Johnson[38] defines knowledge based systems as systems which manipulate 'knowledge' in order to perform a task or tasks. The knowledge in a knowledge base is highly structured symbolic data which represents a model of the relationships between data elements and the uses made of them. Moreover, he defines the field of expert systems as a subgroup of knowledge based systems.

5.2.2 Artificial Intelligence

The following definition from Barr and Feigenbaun (1981), cited by Jackson[40], of artificial intelligence is representative of opinion in the field.

'Artificial Intelligence (AI) is the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit the characteristics we associate with intelligence in human behaviour - understanding language, learning, reasoning, solving problems and so on.'

In other words, AI is concerned with programming computers to perform tasks that are presently done better by humans, because they involve such higher mental processes as perceptual learning, memory organisations and judgmental reasoning. Thus, writing a program to perform complicated statistical calculations would not be seen as an artificial intelligence activity, while writing a program to design experiments to test hypotheses would.

In their publication 'Expert Systems Tools and Applications', aimed at a more practical reader, Harman et al.[42] define artificial intelligence as an academic research program, in the same sense that Physics is, which is aimed at determining what sorts of things computers can be made to do.

Some AI researchers conceptualise their work as an exploration into the nature of human intelligence or recognition, but most are interested in determining how computers can be used to solve specific problems. Besides other lines of application, some AI researchers are concerned in developing computer programs that reason like human experts. In other words[40], they are techniques that allow specific types of expertise to be 'captured' and get the computer to reproduce recommendations or

decisions which would otherwise be made by a human expert. From this branch came expert systems, one of the commercial offsprings of AI research.

5.2.3 Expert Systems

As pointed out by Waterman[37], expert systems are a consequence of a conceptual breakthrough of AI researchers in computer programming, the result of a 20-year quest to define the appropriate nature of programs that could in some sense think, that is, solve problems in a way that would be considered intelligent if done by a human. To reach this point they had to pass through a number of phases. First they tried to simulate thinking by finding general methods for solving broad classes of problems; they used these methods in general-purpose programs. This proved to be too difficult and fruitless. The more classes of problem a single program could handle, the more poorly it seemed to do on any individual problem. They then tried to concentrate instead on developing general methods or techniques to use in more specialised programs, which were techniques like representation - how to formulate the problem so it would be easy to solve - and search - how to cleverly control the search for a solution so it would not take too long or use too much of the computer's memory capacity.

It was only in the late 1970's that workers in this field began to realise something important, which may be stated as:

'To make a program intelligent, provide it with lots of high-quality specific knowledge about some problem area.'

This realisation led to the development of special-purpose computer programs that were expert in some narrow problem area. These programs were called expert systems.

There are several ways of defining expert systems but, although some of them place stress on different aspects, the basic idea is the same.

As defined by Jackson[40], 'an expert system is a computing system capable of representing and reasoning about some knowledge-rich domain, with a view to solving problems and giving advice'. It can be distinguished from other kinds of AI programs in that:

- it deals with subject matter of realistic complexity that normally requires a considerable amount of human expertise;
- it must exhibit high performance in terms of speed and reliability in order to be a useful tool;
- it must be capable of explaining and justifying solutions and recommendations in order to convince the user that its reasoning is, in fact, correct.

According to Michie (1982)[54] an expert system embodies in a computer knowledge base components of an expert skill, in such a form that the system can offer intelligent advice and, on demand, justify its own line of reasoning.

Harman et al.[42] see further advantages of the technique: "In essence, expert systems techniques enable computers to assist people in analysing and solving complex problems that can often be stated only in verbal terms. Thus, they extend the power of the computer beyond the usual mathematical and statistical functions and facilitate the creation of computer programs that conduct various possible courses of action". Moreover they say that 'equally important, some expert systems techniques make the development of sophisticated programs possible for people who lack programming skills'. '... other managers or technicians who also lack programming skills can easily examine the knowledge within the system and revise it when necessary'. This is equivalent to saying that an expert system can capture expertise on a computer and make it available to non-expert users[40]. The fact that knowledge can be examined and modified by experts not skilled in programming comes from the explicit representation the technology provides, as will be seen in due course.

Nevertheless, there are two distinct features in the building process; one of the expert and the other of the 'builder'. Therefore, for the sake of explaining the building process and features that characterise expert systems, we shall separate these two activities as if there were two individuals involved.

The process of building an expert system is often called knowledge engineering[37]. It typically involves a special form of interaction between the expert system builder, called the knowledge engineer, and one or more human experts in some problem area. The knowledge engineer

"extracts" from the human experts their procedures, strategies, and rules-of-thumb for problem solving, and builds this knowledge into the expert system.

Harman et al.[42] go on to say that the three most important ideas that characterise expert systems are: 1) A new way to represent knowledge; 2) heuristic search and 3) the separation of knowledge from inference and control. They define each as follows.

5.2.3.1 Representation of Knowledge

Normally the term knowledge refers to a body of information about a particular topic that is organised to be useful. If it is said an individual is very knowledgeable about a subject, it is assumed the individual not only knows a lot of facts about the subject, but also can use that information to analyse problems and make judgements about related topics.

AI researchers have focused on the verbal and graphic aspects of knowledge rather than the more mathematical aspects that conventional software people have studied. Thus, where a conventional programmer might seek to reduce a problem to elements that can be represented in mathematical terms and manipulated by an algorithm, AI programmers are more interested in knowledge expressed in sentences and pictures and manipulated by logical inferences. Encoding linguistic expressions and manipulating these with logical procedures leads to a major advance in the types of problems that can be solved by computers.

5.2.3.2 Heuristic Search

Conventional computing depends upon a complete analysis of all the elements and steps in a problem. In effect, this limits the domain of conventional computing to problems that can be exhaustively analysed. Humans solve problems that are too large and complex to be understood completely by using heuristics (rules-of-thumb). By their very nature heuristics can lead to errors, but they allow humans to reduce a large problem to one of reasonable size, serving to increase the likelihood of finding a usable answer. Heuristics depend on knowledge of the specific situation, and they are usually acquired from experience. They do not guarantee success, but represent best estimates which are effective in some subset of cases encountered in practical problems.

Any competent professional has a huge body of heuristics to use when faced with a problem that involves uncertainties. Every manager has rules-of-thumb to use when forced to act with a less than complete analysis of a situation.

By using heuristics, AI programs can suggest actions in situations where conventional programs could not. On the other hand, like human advisors in such situations, the new programs will qualify their recommendations, and they will sometimes be wrong.

Heuristic programming techniques have rapidly expanded the variety of tasks computers can tackle. They will allow programmers to develop larger, more complex programs able to analyse very fuzzy problems and

make suggestions. Ultimately, however, heuristics depend on confidence techniques that allow human experts to qualify their judgements. Typically, heuristic based programs do not reach a correct answer, but suggest options and provide some estimate of the likelihood of each being correct.

5.2.3.3 Separating Knowledge from Inference and Control

In conventional programming, knowledge about a problem and procedures for manipulating that knowledge to solve the problem are mixed together. When non-programmers look at the code for a computer application, they cannot begin to understand what the program knows or assumes about the world. This means that the human expert must depend on a program to express that knowledge correctly, making it difficult for other experts to look at the program to see what the developer assumed about the problem.

AI researchers have developed a number of techniques to separate knowledge in a program from procedures to manipulate that knowledge. In effect, any expert can examine the knowledge in an expert system and determine if the knowledge is correct. Moreover, when knowledge about a problem changes, the expert can point to the exact rules or assumptions to change.

The separation of knowledge from inference and control is probably the most important concept to come out of AI research. On the surface it is a simple concept, and it can be just as easily implemented in

conventional languages as in AI languages. Once a programming environment can be created that will "develop its own algorithm" to manipulate a given body of knowledge, anyone who can provide the knowledge can create a program.

5.3 Features that Characterise Expert Systems

5.3.1 A Simple Example

The task of the the knowledge engineer is to produce a knowledge base, composed of rules that might be as simple as the following:

> if the 'traffic light colour' is green[55]
>
> then action is proceed;
>
> if the 'traffic light colour' is red
>
> then action is stop.

Within these simple rules the following components can be seen[55]:

- Logical operation (if ... and ... then).
- Objects to which the logic applies ('traffic light colour' and action).
- Facts or observations which relate to the objects (green, red).

This example helps explain some important characteristics of expert systems.

5.3.2 General Features

These general features that characterise an expert systems are[55]:

- It processes a defined set of rules and facts to represent expertise.
- It is designed to grow on an evolutionary basis, improving its 'expertise' as it grows. It can be easily seen that in such a structure new rules on the subject can be added without upsetting the proceeding rules and that the program would then still work (e.g., we could include 'traffic light colour' as amber). Conversely, if rules are removed, it may also continue to work.
- It can explain its reasoning in an understandable way. Expert systems are structured in a way to allow the ability to retrace a set of rules and explain how and why a recommendation was derived, simulating this human characteristic to justify decisions.
- They are limited to a specific area of human expertise.

Because expert systems must deal with data as well as rules they commonly make use of devices called 'frames'. A frame is a way of holding data on some subject in a tightly bundled group that can be referred to as a single entity. Each datum within the group is held in what is often referred to as a slot. Note that the data handled can be textual or numeric in character.

5.3.3 Knowledge Base and Inference Engine

The codified knowledge of an expert system is known as the knowledge base. It comprises a complete set of rules and frames describing a topic. As well as the knowledge base, an expert system needs to be able to process the knowledge and have the ability to 'reason' in the sense seen in Section 5.2. The reasoning part of the expert system is called the inference engine[55]. The inference engine is the part of the knowledge base system or expert system that contains the general problem-solving knowledge[37]. It may well be a very complex program but its detailed workings do not affect the users understanding of the knowledge being applied.

The inference engine uses the knowledge base rules in logical sequences, which need not be ordered in the coding of the knowledge base. This rule processing is done by using two major control strategies, i.e., Forward Chaining and Backward Chaining[54].

These strategies are combined with a structure[41], which determines the way in which the various rules are applied. These are called depth-first search and breath-first search, see Figures 5.1a and b. In depth-first searches the left-most branch of a tree is followed first until either the query is solved or the end of the branch reached. In the latter case it then "backtracks" to the last choice point and examines the alternative branches in the same way[54]. In the

breadth-first searches, the search is layer by layer through successive levels of the search space[40]. Thus in Figures 5.1a and b the processing of the search would follow the numbered steps.

Breath-first searches find the shortest solution path, if there is one, but depth-first searches find a solution faster as long as they are guided in some way, i.e., if they make good decisions when choosing which path to pursue next. On the other hand, depth-first searches may never terminate if the search space is infinite, even if a solution exists along some as yet unexplored path. Guided depth-first searches are therefore preferred; this is called the 'best-first search', since at each choice point, it is trying to make the best decision as to where to go next.

In Forward Chaining the left hand side of all rules are matched against the data. From those that match, a rule is selected and the right hand side of the rule is executed. This process is repeated until a whole cycle is executed, i.e., all rules that can be tried, are tried in sequence, according to the logic programmed[54].

Backward chaining is goal oriented. In backward chaining an initial goal is selected on the right hand side of the rule and if the left hand side of goal rule matches the data, the process ends successfully. If there is no match or a partial match, the inference engine looks for rules with a right hand side which matches those clauses that were unsatisfied and uses these as new sub-goals, repeating the process until the goal is reached[54].

These two control strategies suit different applications: for selection applications forward chaining is more efficient; while for diagnostic applications it is better to have a goal driven method, i.e., backward chaining[54].

5.4 Typical Areas of Application of Expert Systems

As can be seen by a brief study of the AI literature, a great number of areas can benefit from the application of expert systems. Hayes-Roth et al.[39] classify the types of expert systems that can be developed from knowledge engineering as follows:

Category	Problem Addressed	Example
Interpretation	Inferring situation description from observables.	speech analysis
Prediction	Inferring likely consequences of given situations	weather, military forecasting
Diagnosis	Inferring system malfunctions from observables	medical, software
Monitoring	Comparing observations to plan for vulnerabilities	plant operation
Debugging	Prescribing remedies for malfunctions	software
Repair	Executing a plan to administer a prescribed remedy	automatic maintenance
Instruction	Diagnosing, debugging, and repairing student behaviour	tutorial
Control	Interpreting, predicting, repairing and monitoring system behaviours	air traffic control
Design	Configurating objects under constraints	circuit layout, budgeting

Planning	Designing actions	robots, military planning problems

Various companies and research centres have applied these techniques to a number of areas such as[37]: chemistry, computer systems, electronics, engineering, geology, medicine, military, etc. The number of applications in these areas is quite vast. A few of the better known or wide ranging applications will be mentioned here:

One of the most successful systems developed is called XCON, which is a design application for computer systems. Its purpose is to configure VAX 11/780 computer systems (it is also known as R1[41,42]). From a customer's order it decides what components must be added to produce a complete operational system and determines the spatial relationships among the components. XCON then outputs a set of diagrams indicating these spatial relationships to technicians who assemble the VAX system. It handles the configuration task by applying knowledge of the constraints on component relationship to standard procedures for configuring computers. The system is non-interactive, rule-based, and uses a forward chaining control scheme. XCON is implemented in OP55 and was developed through a collaboration between researchers at Carnegie-Mellon University and Digital Equipment Corporation (DEC) in Hudson, Massachusetts, U.S.A. This commercial expert system configures VAX computers on a daily basis for DEC and is the largest and most mature rule-based expert system in operation[37].

Another very sophisticated example of expert systems featuring rule generation are the systems DENDRAL and META-DENDRAL, chemistry applications developed at Stanford University. DENDRAL infers a compound's molecular structure from mass spectral and nuclear response data[37]. It is a program which uses a set of rules to reason about the domain of mass spectrometry. META-DENDRAL is a program which reasons about the rules that DENDRAL uses to perform this task. There is clearly a distinction to be made between reasoning with rules and reasoning about the rules one reasons with. This latter idea is an example of what is usually called 'meta-level reasoning'[40]. META-DENDRAL helps chemists determine the dependence of mass spectrometric fragmentation on substructural features. It does this by discovering fragmentation rules for given classes of molecules. The system derives these rules from training instances consisting of sets of molecules with known three-dimensional structures and mass spectra. META-DENDRAL first generates a set of highly specific rules which account for a single fragmentation process in a particular molecule. Then it uses the training examples to generalise these rules. Finally, the system re-examines the rules to remove redundant or incorrect rules[37].

In engineering and correlated fields several major applications can be found[37]. One such system is REACTOR, which helps operators diagnose and treat nuclear reaction accidents, performing interpretation, diagnosis/debugging and monitoring. DELTA is a diagnosis/debugging system which assists in the identification and correction of malfunctions in locomotives. In manufacturing engineering[42] there are

diagnostic systems such as EXPERT EXECUTING, which integrates technology code for aerospace vehicle designers' use, or BRUSH DESIGNER a configuring system to help DELCO engineers design brushes for electric motors. In the Transportation[42] area there is an application named SAFETY of LIFE AT SEA, which helps government inspectors inspect for proper communications equipment on vessels in Canadian waters.

Another natural development that has been emerging from these new techniques is the coupling of symbolic and numerical computing. Kowalik[43] has edited a compilation of several methods and applications, one of which, 'Reasoning about Quantitative Methods in Engineering Design' by Simmons and Dixon, is an attempt to combine conventional CAD representation methods, the analysis methods of computer-aided engineering, and the reasoned application of the expert knowledge of engineers.

5.5 Expert Systems and Ship Design

Progress in applying expert systems to ship design has been discussed in the introduction. Given the capabilities of expert systems noted in the previous sections and studying those used in the field of ship design, it was noted that the ship design system described in Chapters 2 and 4 could take advantage of expert systems features at two distinct levels. First the system shown in Chapter 4, could be contained within a global expert system shell or environment, so that the synthesis process could be controlled by a knowledge base that would either interact with the user or run automatically. This is equivalent to substituting the

data-base handler GENDAT by a knowledge base that would perform exactly the same function, but would additionally allow for some synthesis decisions to be taken heuristically and contribute a better understanding of the whole process by backchaining to interrelate the various disciplines involved in the process. The second level identified, where expert systems may be more readily applied is in specific narrow domains, such as optimization control, an area where expertise is required but the user need not be an expert, with the expert system forming a component of the main system.

Writing an expert system at this first level would clearly be a major undertaking spanning many years of effort. However, some work has been done on the development of a data-base handler using the commercially available Expert Systems package 'Leonardo'[55] taking advantage of its particular features. The program developed is just for test purposes where a few manipulation features with variables and external programs are exercised, but it was developed enough to see that this is a feasible application. It was found that some applications of expert systems would be useful for this work and future developments of it. Leonardo, which is introduced in Section 6.2, is one of the commercial expert systems building tools available on the market and includes all the relevant features of expert systems. The version used is written to run under the DOS operating system and therefore is incompatible with the current ship design system, which uses UNIX. For this reason fully

integrated tests were still not possible. Nevertheless, useful tests and examples were developed on a separate computer, aiming for a full integration with the system in the near future.

To investigate applying expert systems to more narrow domains, a number of test programs were developed. The following sections deal with these example applications for specific tasks, and some aspects give insight on how the completely integrated suite would look.

5.5.1 Managing Specification (Input) Data Uncertainty

As has been noted in Section 4.1.2, the input data for the design system is a fixed list of parameters (e.g., length to displaced volume ratio, beam to draught ratio, etc.), and these are used to start the design process. It was mentioned there, that the hierarchy of the rule based structure in-built in the routine CONSST generates parameters derived from this list that could be easily changed for designs using different combinations, such as a given length and beam, say. It was also noted that it would be useful to have this done without having to change the code of the CONSST routine every time different initial parameters formed part of the specification.

Using the Leonardo expert system, a simplified example was exercised to tackle this problem: the idea of the program developed was to utilise given data to calculate those missing, in any combination, as long as all the required data was supplied, that is, the process was to put a

certain parameter in evidence. For simplification, the expression $\nabla = L \times B \times T \times C_B$ was used and a small set of rules in a knowledge base was created to calculate any missing parameter given any other four.

This was achieved through the use of 'flags' as text objects and 'instanciations' as numerical objects, enabling the sequence of 'firing' the rules of the inference engine to follow different sequences according to which parameter was left out at each run. Developing this small program helped exercise the capacity of an expert systems program to compute 'first what it can', using forward chaining, regardless of the sequential order they were originally programmed in.

This process is programmable in languages such as Fortran, as any other process would be, but this would require a very difficult and tedious effort, since the logic is cumbersome to deal with. This is precisely one of the advantages of separating the knowledge base from the inference engine, with the latter performing a forward chaining strategy. This exercise gave insight on how such a process might be programmed in Fortran. In fact, this information has led to a subsequent implementation of the process in the CONSST routine for the relations normally required by the design system.

5.5.2 Example of Coupling Symbolic and Numeric Relations - Interdependency of Design Parameters

To further develop insight into expert systems, two more calculations were added to this test program: wetted surface area and metacentric height. As with the displacement calculation, these were very straight forward, using simple formulae[18] to demonstrate instanciation of values, forward and backward chaining and the use of object frames for formula procedures and display of results (layouts). Again, the idea of setting flags as text objects with names such as 'transverse-inertia', 'centre-buoyancy', etc., was used to find the interdependency of parameters. In other words, the decision trees of each calculation could be accessed to work out which parameters depended on others.

A schematic view of the interrelations of parameters as programmed can be seen in Figure 5.2. This structure was programmed in a way that allows heuristic decisions to be made on what to look at next (e.g., if KB, BM and KG are known, run STABILITY to find GM; stability is then known), while procedures are called to perform the calculations (e.g., run STABILITY implies GM = KB + BM - KG). Numerical calculations are thus done by forward chaining and the interrelations or interdependencies can be established with both forward and backward chaining.

When explanation devices are used (i.e., 'WHY' and 'HOW' devices), the relevant rules are shown, in the first case explaining why a rule needs data and in the second, which occurs after a calculation, the rules are displayed in reverse order making interdependencies of the various parameters clear for the user to inspect.

5.5.3 Controlling Optimization

Expert systems can provide very good self explaining, backward chaining, goal oriented problem solvers. The application of a knowledge base to control optimization is an example of control over design techniques and can be extremely useful because it allows naval architects to take advantage of the technique without having to be experts in optimization, leaving their time free for ship design considerations. The need for this application has been shown in Chapters 3 and 4 and its development and full application to a frigate design are discussed in Chapter 6.

5.6 Limitations of the Technology

Although there are a number of applications of expert systems to various areas of knowledge, these are mainly restricted to problems that deal with facts rather than large scale calculations. This is natural, since expert systems were designed to solve symbolic, non-numerical problems where there was lack of programming capabilities. Therefore, it should not be surprising that they are not commonly applied to engineering design problems. However, this use is increasing where the engineering

problems are specific to narrow domains of knowledge. Nonetheless, in a broader sense, for complex engineering system designs, very few attempts are being made.

The design process is an activity where creativity and innovation are desirable and sometimes essential. This is in contrast to most areas where expert systems have been applied. Consider medical diagnosis systems, for instance, where one wants to identify a disease and eradicate it. More knowledge of the subject only adds to the procedures and solutions and it is very rare to have a change in logical reasoning about the subject. In design or engineering design what happens is, in many ways, the opposite: the interrelations in the design process are such that the logic must be questioned at times, because it is creating something new. For such, one must find ways to break rules in a sensible fashion to satisfy new, ever changing requirements. The process involves many compromises and previous lines of reasoning may not apply. Consequently, if any of them are consolidated into the design method, this may jeopardise innovation.

By contrast, Chapter 3 mentioned that systems structured for optimization can allow for goal oriented tasks, where targets and restrictions can be specified and improvements driven by measures of merit. They can perform exploratory searches, that is, examine competing ideas and search for successful combinations. In such structures innovation is more likely because they test previously untried combinations of parameters and creativity is possible if they allow for modifications of the way analytical tools are used.

Expert systems can provide great help in logical programming terms, but they still cannot define an idea or good solution from a combination of, say, two others. The decision trees of expert systems are still hard coded. This is recognised by AI researchers. There has been some effort in this direction, such as the system META-DENDRAL mentioned earlier. Waterman[37] gives some insight of the current state of art of this ability: 'explanation is just one small aspect of self-knowledge. In the future self-knowledge will allow expert systems to do even more. They will be able to create the rationale behind individual rules by reasoning from first principles. They will tailor their explanations to fit the requirements of their audience. They will be able to change their own internal structure through rule correction, knowledge-base reorganisation, and system reconfiguration. A first step in this direction is to make the expert system's meta knowledge separate and explicit, just as the domain knowledge is now made separate and explicit.'

When comparing expert systems with human experts, Waterman[37] gives us further insight of these difficulties: 'Another area where human expertise excels is learning. Human experts adapt to changing conditions; they adjust their strategies to conform to new situations. Expert systems are not particularly adept at learning new concepts or rules, probably because this is a very difficult task that has always been somewhat of a stumbling block for AI. Progress has been made in

developing programs that learn, but these programs tend to work in extremely simple domains and do not do well when confronted with the complexity and detail of real-world problems.'

Thus, expert systems cannot search for new combinations of ideas as optimization processes can. Such ideas are best evaluated through numerical interpretation and expert systems are not designed for that, they cannot do it. The way out of this for expert systems is further development of meta-logic capabilities, which will allow for a real revolution in programming.

In consequence, expert systems can jeopardise creativity if not well used. Creativity is still a very human ability and we are still a long way from being able to simulate it by any means. Waterman[37] has a few words about this from the AI point of view, when mentioning limitations of the technique: 'one such area is creativity. People are much more creative and innovative than even the smartest programs. A human expert can reorganise information and use it to synthesise new knowledge, while an expert system tends to behave in a somewhat uninspired, routine manner. Human experts handle unexpected events by using imaginative and novel approaches to problem solving, including drawing analogies to situations in completely different problem domains. Programs have had little success doing this.' The implication seems to be that if creativity is desirable or required in design, design systems must be developed in a way that would allow for it. One way of doing this is to

structure a system for modularity of computer routines or design theory modules, so that new ideas can be programmed, inserted or modified in a rapid and practical manner.

If a piece of theory is broken up to become part of a knowledge base (as in the displacement equation of Section 5.5.2), although the ability to justify each step is gained and the knowledge is made more explicit and easier to program and to understand, there are several adverse consequences:

As they become part of the main stream of the programming code, which is the knowledge base, different levels of knowledge get mixed. If one is to work with a system that uses both fundamental but simple synthesis, decision-making and detailed calculations the knowledge base may look confusing and unbalanced, losing its universality and putting irrelevant rules into the justification stream. There could be an introduction of mathematical functions into the knowledge base (for procedures to be called by a rule) which cannot be explained to the user like rules since they are not logical but numerical relationships. The way around this would be to have knowledge sub-bases which would make the expert system application no longer domain specific, which in fact seems to be the case of ship design from the point of view of the naval architect.

Consider the example of minimising resistance when constrained by stability. If the parameter available to the designer were beam to depth ratio the natural instinct would be to increase the ratio when stability was insufficient and so this might be programmed as a rule in

the knowledge base. However consider hulls where the ratio is around unity. For such hulls, DECREASING the ratio might make them more stable because the centre of buoyancy drops faster than the metacentre. If instead of a general rule a detailed and accurate calculation can be substituted this is clearly better even if the programme is not able to give a clear explanation for its choices.

In other words, if the design theory is part of the knowledge base the system will be narrow in scope and detailed in knowledge. Such a structure will be able to give reasoning about the knowledge but it may need to simplify the knowledge itself. It will also constrain the design, rather than the techniques that control it, making it easy to correct some decisions, but hard to change the theory.

On the other hand, if the analytical theory is kept out of the knowledge base instead, being called as procedures when needed, the knowledge base becomes more universal and interdisciplinary, the scope becomes broader, allowing for higher level decisions, better application and control tools, the system gains modularity and the knowledge can be even more detailed. Nevertheless, there will be no reasoning about the inner workings of each piece of theory; it would be a design synthesis expert system. In such a structure goal oriented tasks are possible and the synthesis process can be result oriented.

The characteristics of expert systems compared with those of optimizers seem to suggest that a combination of both technologies might be useful in a design system. The idea being to allow powerful techniques to be

employed by non-expert users while stopping short of placing detailed knowledge of the relevant design theory in a rule-base, which would be difficult to produce or merge, i.e., to have a knowledge base on HOW to use design theory, not on the theory itself. Since the most obscure area of theory presented in the previous chapters concerned optimization this seems to be the natural target for applying expert systems techniques. Such an application is described in the next chapter.

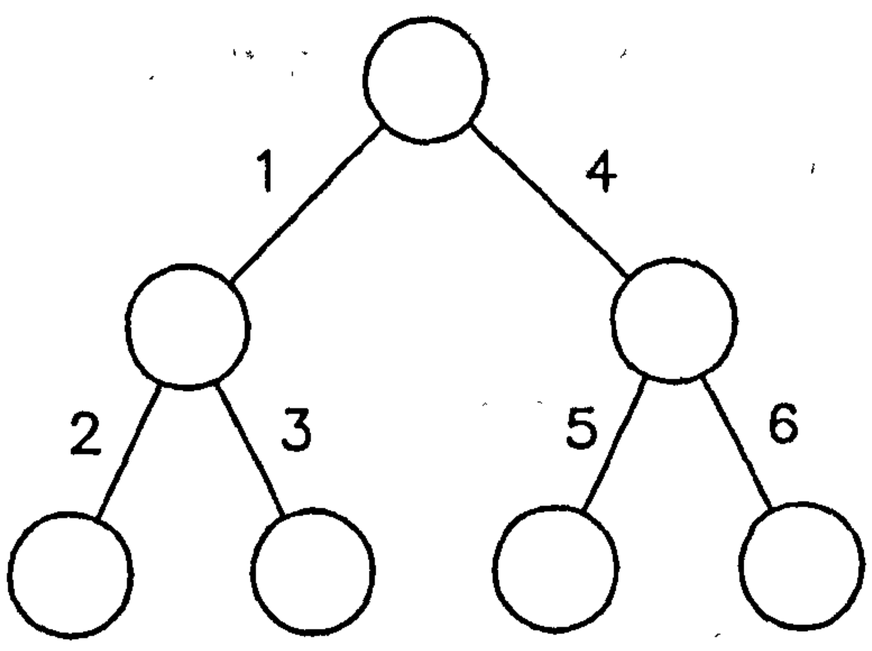

Figure 5.1a – Depth–first search

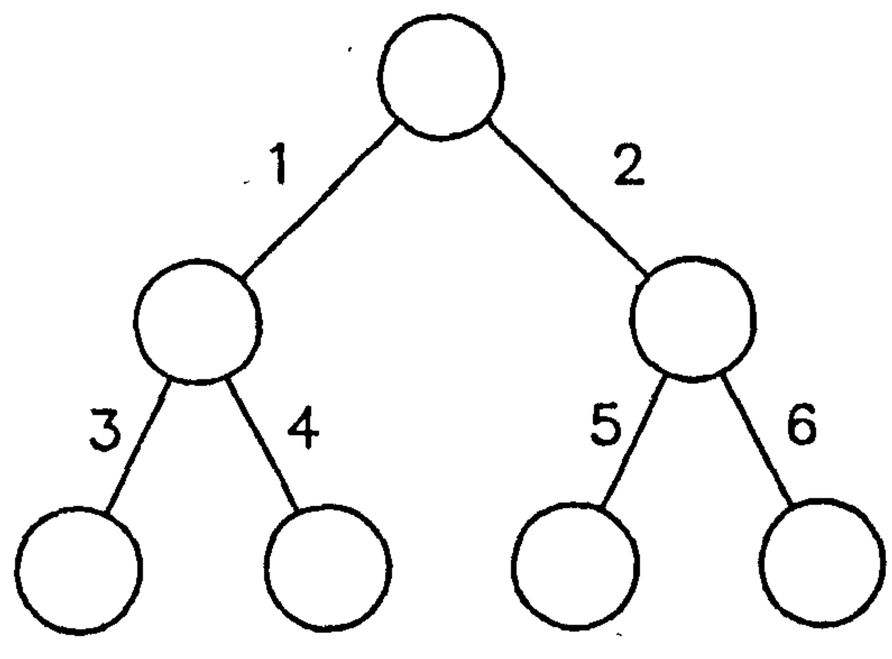

Figure 5.1b – Breadth–first search

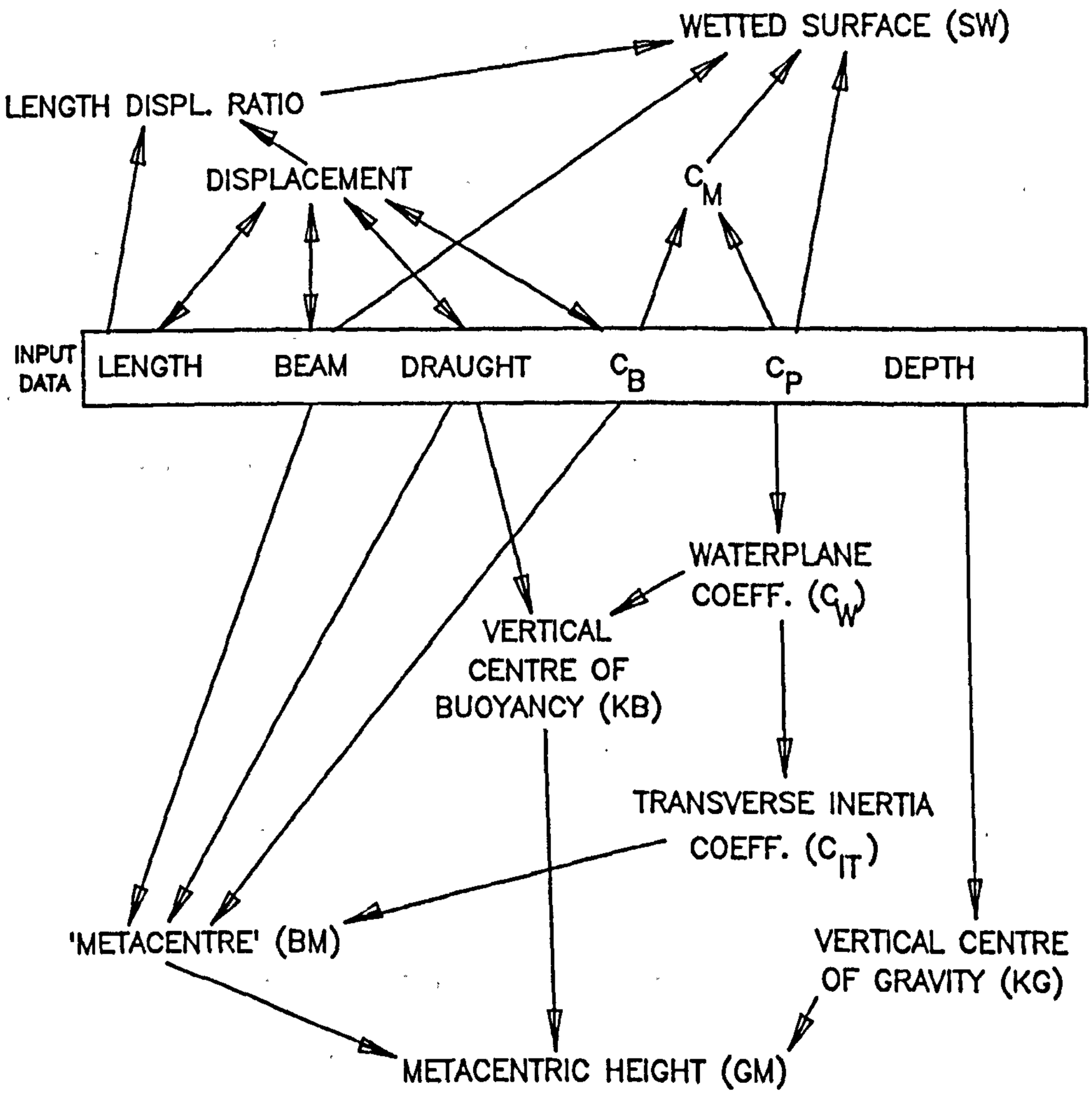

Figure 5.2 – Schematic view of an example–test program to relate parameters using an expert system shell

6. AN EXAMPLE APPLICATION OF A KNOWLEDGE BASE TO CONTROL OPTIMIZATION IN THE SHIP CONCEPT DESIGN SYSTEM

6.1 Ship Design optimization problems and control

The example given in Chapter 4 was obviously a comparison exercise for the various optimizers, reinforcing the suitability of the methods chosen for the problem studied and also confirming which ones were fastest and the inverse relationship of robustness. For these aspects it seems to be obvious that, if the design problem can be solved with a method like APPROX, say, this is better than running it with a slower one. It also seems logical to think that if the mapping strategy is enough to show the region where the optimum can be found within reasonable engineering precision, optimization methods would need not to be deployed. From the analysis of Section 4.2.5, it can be seen that, in real cases, there is no point in running different optimization methods from the same starting point. Nevertheless, using optimizers is by no means straightforward and so in order to control the optimization process, a knowledge based structure has been developed, using the expert-systems builder's shell, Leonardo[55].

The goals to be achieved with the knowledge base controlling the optimization process would be: (1) to find the true minimum or reasonable results within engineering precision; (2) to do so with the lowest possible computing time (defined by the number of design loops), and (3) to ensure trustworthy answers.

6.2 The Leonardo System

Leonardo is an expert-system building tool where the developer programmes the knowledge base, i.e., the rule base, frames and slots, but the inference engine is preprogrammed. The frames can be programmed as object frames or procedures, the latter allowing for complex programming and links for running external programs. Creative Logic's Technical Summary[55] defines, its basic architecture by noting that it: 'is an object structured system. The basic element of the knowledge base is an object, which may be a rule, a set of rules, a procedure, a specific component of the application such as a record layout, a form design or a dialogue definition. Structure rotates around the object directory, which allows access to any of the objects in the knowledge base, provides cross-reference amongst the objects, and provides a self-documenting architecture for the application. The expanded definition of an object is held in a frame. Leonardo has implemented class and member frames, and supports inheritance and quantification over the frame base. These are typical features of a full knowledge engineering environment. As is normal with such programs, the frame associated with an object provides the information necessary to derive a value for the object when needed. In particular, procedural attachment is supported, using the internal procedural language of the system, or access to external procedures and processes'.

To explain the Inference Strategy of Leonardo, Creative Logic's[55] Technical Summary may be quoted again:

'The basic approach of the inference strategy employed is an exhaustive, left-side, depth-first backwards search. However, when any object is instanciated, the system performs an unbounded breadth-first forward search, thus maximising the value of the new datum at the earliest possible moment.'

'Another important feature of the Leonardo inference strategy is the structuring of very large rule sets. This is possible due to the frame architecture of the overall system. Any object may own a frame, and in that frame there may exist a slot called a Rule Set: the rule set contains all the rules pertaining to the 'owning' object. This provides three significant benefits:

1. The physical knowledge base has greater clarity, leading to easier maintenance and better documentation.

2. The focus of attention of the system is decided at runtime using the relevant section of the rule base, leading to improved runtime performance.

3. The scheduling of tasks with respect to the next rule set to be executed and the stacking of outstanding rule sets falls naturally out of the semantics of the rules themselves, without the builder having to write control directives into the systems.'

'The inference strategy is modifiable by the builder. Forward chaining may be switched on and off, both locally and globally. Alternatively, the system may effectively be set to run in a pure forward chaining mode, supporting problems addressing synthesis (configuration, planning) rather than analysis (structured selection, fault diagnosis).'

'In addition, the system may be set up to run cyclically. This very powerful feature means that after a goal has been achieved, the system will automatically restart. The builder can determine which values of objects are to be preserved across the reset and restart. This is supportive of in-line monitoring applications, and more complex planning applications. Termination of cycling can be controlled dynamically or statically.'

This system requires approximately 2 MB of hard disk and 512 kB of memory to run. The version in use is a PC compatible one, running under DOS, therefore incompatible with the previously described ship design system, written using the UNIX operating system. For this reason consultations of the optimization knowledge base are made by running two different computers with the advice given by the PC based knowledge base used to control the Sun workstations and the resulting optimization run messages transferred back to the adviser.

6.3 Optimization Control using Leonardo - Knowledge Bases 1 and 2

6.3.1 Knowledge Base 1

Knowledge Base 1 was the first rule base written to control the optimization process using the Leonardo shell. Conceived as a general structure to be tested, and to be subject to modifications and developments, it did not include all modes of failure of the optimization methods. It is a general structure with simple rules and is illustrated as a flow-chart in Figure 6.1. The development of Knowledge Base 1 was based on the behaviour of the ship design function and the optimization process as observed in the design example given in Chapter 4 and, in particular from the analysis of Section 4.2.5, as well as from other minor specific optimization examples taken from studies of a generalised optimization process in design. At a very early stage it was decided to use the objects frames as advice screens and procedures for method selection, assuming that the various external programs would form part of an integrated system including the knowledge base. This would take the optimizers' output, modify features and take further action in the decision making process. It was also noticed, at an early stage, that it would be necessary to use the cyclic capabilities of the shell to run the same rules repeatedly with modified features being accumulated from each optimization attempt.

The basic assumptions for Knowledge Base 1, that were not altered as the Knowledge Base developed, are as follows :-

(1) once selected, the optimization methods are left to run without interruption until they succeed or fail. Judgement and advice on subsequent action to take is based on the optimization method messages given after the runs.

(2) the process would use the methods applied throughout the work, as introduced in Chapters 3 and 4. That is Linear Approximation - APPROX, the Hooke and Jeeves Direct search method - SEEK with two penalty functions and the Random search with shrinkage - RANDOM.

(3) the choice of the method is included in the advisory structure following a hierarchy, starting from the fastest but more fragile ones, using their capabilities until exhausted before going to the next, which show increasing robustness and reliability, but are also more time consuming. Some exceptions were made by applying local, short, random searches at times either to help in clarifying false optima or to help a method being used to find a solution when problems arise.

(4) the mapping strategy, found to be crucial to the optimization problem in earlier chapters, forms the basis of the advisory system. Therefore the process starts from a two dimensional search and progresses to an n-dimensional one, once confidence in and knowledge of the function are acquired.

In the first attempts to carry out the optimization using Knowledge Base 1 it was found that improvements to the basic structure were necessary, and their implementations led to the evolution of Knowledge Base 2. It was found that although the original structure seemed acceptable, the failure modes were not handled well needing too much explanation and experience to understand, i.e., they needed to be broken into more frames and rules for further explanations, advice and decisions. Secondly, it was found that the advantages of saving design improvements to allow restarts from a better point were useful, but this practice was only used in few cases, and this should be extended to be applied whenever possible.

Thirdly, in the case of success, Knowledge Base 1 would give only a crude explanation of how to check for false optima, and if the user was not confident, the advice was to select boundaries around doubtful areas and perform local short random searches, which subsequent testing proved to be impractical. Finally, it was found that the hierarchical selection of methods as initially programmed, when used in the cyclic mode, would not perform properly due to the needs of different approaches for different methods.

6.3.2 Knowledge Base 2

This Knowledge Base was developed from Knowledge Base 1 and is illustrated as a flow-chart in Figure 6.2. The listing of the rule base and the object frames/procedures can be found in Appendix B, where the rules are numbered in accordance with the figure. Every time this

knowledge base cycles back to restart the method in use or to select a different method, different rules are used, according to the consequences of what has been tried before. Judgements are also made based on the success or failure (and modes of failure) of the optimization runs. The same mapping strategy still applies, viz., runs start from a two dimensional problem and after sorting out false optima, the process is then n-dimensionalised. This rather more sophisticated rule base will be described in detail in the following sub-sections.

6.3.2.1 Initialising - Mapping Strategy

The advisor starts by giving instructions on how to select the variables, constraints, limits and the objective function to be used; it gives advice on runs to establish ranges and to avoid inconsistent hull-forms in the process, following the major steps of the mapping strategy outlined in Section 3.5. It goes on to advise on the reduction to two variables and the use of Simpleplot to prepare the relevant mappings for the problem in hand, to enable the user to acquire a notion of what to expect of the two-dimensional problem and to develop confidence in the behaviour of the function in the n-dimensional process to come. The advisor then suggests running two-dimensional applications of the optimization methods following the hierarchical sequence mentioned previously.

6.3.2.2 The "Success" Branch

When an optimization run indicates that it has been successful, the Knowledge Base will ask the user to check for the existence of false optima (see lower branch of Figure 6.2 and rules numbers 25 to 30). Unlike Knowledge Base 1, no optimization runs are required at this stage. The situations predicted are such that the user would either check visually his original maps, or perform a few extra local ship designs for his mappings, or do so from eventual amplifications of certain smaller doubtful areas. Once any saddle points, false optima or uncertainty in the behaviour of the function is clarified, the user will be advised to n-dimensionalise the problem and run from the last method in use. Advice is also given on what to type into or modify in the data-base handler. If there is a successful outcome of optimization of a n-dimensional run, the advisory session ends declaring that result as the final optimum.

6.3.2.3 The "Failure" Branch

In the case of an optimization run failing to reach an optimum, the reason for this failure will be addressed. In Knowledge Base 2 all modes of failure for each of the methods used from OPTIVAR are included in the appropriate menus. Also, these modes are tackled with a more complex approach and in more detail than in the previous Knowledge Base. Whenever suitable, advice is also given to save the last, best, results achieved before proceeding to corrective action. This is in order to save optimization time, while retaining improvements for

subsequent use. For each advice given there are appropriate frames, with a paragraph or more, of comments and explanations of the probable behaviour of the method in the function. The advisor may some times decide to abandon a certain method, skipping to the next one, either automatically, based on a judgement of the circumstances, or after asking the user about the sort of function he believes is being examined. In such cases explanation is also provided. Also, whenever advice on corrective action is given, there is guidance on what and how to modify ship parameters and what to modify of the optimization method's features, according to that circumstance. Of course, such advice tends to be design specific and suffers from the drawbacks noted in the previous chapter but it is limited to a great extent here.

Due to the relationship between speed and reliability of the various methods, the idea of trying each method until exhausted before going to the next one was retained. The corrective actions suggested in the rule base aim for objectiveness and short run times in, supposedly, worthwhile areas. These were based on the results of tests and studies of the different peculiarities of each method (see Chapter 3), including advice adapted from the OPTIVAR package and messages.

APPROX

The left branch of the flow-chart of Figure 6.2 caters for the modes of failure and corrective action of APPROX. Failures occur if the starting point is infeasible (rule No. 5) or if there is no convergence (rules 6-11). When there is no convergence the rule base suggests in turn closer starting points (rule No. 7), changes to step sizes (rule No. 8)

or, as a last resort, an isolated random search before resuming with APPROX (rules 9 and 10). This last approach is warranted because, when it can be used, APPROX is the fastest method available. If none of these are successful (rule No. 11) or the function cannot be linearised successfully (rule No. 6), the method used is changed to SEEK with OPTIM 1.

SEEK and RANDOM

If, after reasonable attempts to obtain a final result using SEEK with OPTIM 1 fails, it goes on to SEEK with OPTIM 2. If all fails, a full RANDOM search is advised as a last resort and the user is warned about time consumption and also to use the shrinkage factor with care. A brief comment is made on the type of function and constraints that the user might be dealing with.

The right branch of the flow-chart in Figure 6.2 caters for the modes of failure and corrective action of SEEK with OPTIM 1 first and then with OPTIM 2. Although the two penalty functions have different behaviours, they fail for similar reasons. Here, corrective action is again taken based on the previously mentioned studies, and in some cases a retry with a different starting point is worthwhile (see rules No.13 and No.19). Finally, as observed for the cases mentioned in Chapter 4, the possibility of shifting limits of variables to newly unpredicted combinations together with a relaxation of the constraints (rules 16 and 22) is sometimes advised to achieve better results.

6.3.2.4 Using the 'Why' and 'How' Devices and Backchaining

Within Knowledge Base 2 the 'why' and 'how' devices of the inference engine can be used. The 'why' device will display a rule to explain the reason for a question at any time and it is possible to backtrack from this to previous rules. This only stops when it reaches the choice of method because at this point the rule base is cyclic. This cycling of the rule base is extremely useful, allowing a great number of steps to be programmed in few rules, but a consequence is that it over-writes certain rules, since it uses them more than once.

The 'how' device, which is used at the end of a session, can backtrack up to two methods (or cycles) showing all the rules used in between. Finally, useful backchaining of the knowledge base can be carried out. These are such as object = 'false-optima'; value = 'sorted-out' will give the rule where the conditions of checking for false optima are listed and can be in turn backchained.

6.4 Example Testing of Optimization Control using Knowledge Base 2

One way to test Knowledge Base 2 would, of course, be to run the example of Chapter 4, following the advice given by the rules and comparing the process with the manually controlled optimization one. The outcome could be summarised as follows:-

(1) if a two-dimensional run had been carried out with (m) and $B_{wr.}/T$ as variables, using the APPROX method, after 24 loops it would have reached the results shown in Table 4.3 of Chapter 4.

(2) from this successful result a five-dimensional run would be required (after checking the mappings for false optima) and APPROX would fail. If all subsequent attempts to take corrective action to make APPROX succeed also failed, the advisor would then suggest the run of SEEK with OPTIM 1.

(3) SEEK with OPTIM 1 would be successful in few runs, repeating the same run as shown in Table 4.4 of Chapter 4 from a much closer starting point. It would not suffer any hang ups, and it uses one local random search of 100 shots as built into the method, see Section 4.2.4.

It can be seen from this brief example that this application of the Knowledge Base does not substantially decrease the time taken in the optimization processes. However, combined with the mapping strategy, it would increase confidence in its usage and would also increase the process efficiency.

Nevertheless, such an example is not enough to provide a good test of the Knowledge Base capabilities. Therefore, it was decided to make a few modifications both to the problem constraints and also to the initial ship to create artificial difficulties to allow the process to explore and test several situations and observe if and how they are overcome and what improvements to Knowledge Base 2 would eventually be required in consequence. For the sake of comparison, the ship and the theory are taken to be as close as possible to the previous example, i.e., only slight modifications are made.

6.4.1 Modifications to the Constraints - Mappings

It was found that to make the knowledge base deploy more of its capabilities it was necessary to:-

(1) give a greater weight to variables other than (m) and B_{WL}/T, (which proved to be too dominant), and therefore to move the final optimum further away from that found from the two-dimensional search.

(2) make different constraints act at small variations of design parameters by making them nearly coincidental at the region where the optimum lies. This tends to produce edges which can produce hang ups in direct searches and force the methods' analysis strategies to investigate different constraint violations at small variations in the design parameters.

To achieve this, the minimum enclosed volume required (which is one of the constraints) was set at a very high value to force the design process to search for a bigger and/or highly flared ship. Based on a number of manually prepared mappings, the stability constraints were also made artificially more severe to force the search to place more emphasis on a third design variable. The stability criteria for the GM requirement was increased to 1.0 m and the areas under the stability curve between 0 to 30 deg., 30 to 40 deg. and 0 to 40 deg. were made more severe by being set three times greater than the original I.M.O. values, see Table 6.2. Although not reflecting a realistic stability

requirement, these values achieve the previously mentioned coincidence of the stability constraints in a "would be" optimum region at high flares. With these changes the enclosed volume restriction became the dominant lower constraint limit at lower flares, while the stability constraints took over at regions of high flares, allowing for ships with less resistance in these regions. Figures 6.3a and b and 6.4a and b, illustrate this behaviour. Figures 6.3a and 6.4a show the design function of resistance for (m) and B_{WL}/T in the same ranges used in Chapter 4 without the action of the constraints. In Figure 6.3a the flare is zero (vertical sided) and in 6.4a the flare is at the extreme limit of feasible hull-forms. Figures 6.3b and 6.4b show the function as constrained for $FLARE_x$ = 0 deg. and $FLARE_x$ = 17.1 deg., respectively. As can be seen, the lower limit restriction of the vertical sided designs (6.3b) is the enclosed volume, while the lower limit of the highly flared ships (6.4b) is constrained by stability. It can also be noted that for $FLARE_x$ = 0 deg. (6.3b) the enclosed volume constraint is much more severe than the stability constraint at $FLARE_x$ = 17.1 deg. (6.4b), not allowing for lower B_{WL}/T ratios to provide lower resistances. This dominance of one constraint over the others can be better illustrated by the more detailed curves shown in Figures 6.5a, b and c and 6.6 a, b and c. Figures 6.5 represent $FLARE_x$ = 0 deg., while figures 6.6 represent $FLARE_x$ = 17.1 deg., all being amplifications of figures 6.3 and 6.4, but showing the constraints acting independently. In Figure 6.5c the clear dominance of the enclosed volume over the stability curve areas (6.5b) and the GM (6.5a) can be seen, permitting only B_{WL}/T ratios over 4.2. In the highly flared case (Figures 6.6), the enclosed volume limit occurs below B_{WL}/T of 2.5 (6.6c), while the stability curve areas (6.6b) and the GM (6.6a) act at B_{WL}/T ratios above

3.2. The results of making the stability constraints nearly coincidental can be seen by comparing Figures 6.6a and 6.6b where the dominance of one over the others changes with (m) variations.

6.4.2 Modifications to the Initial Ship

To gain the most from this search it was found that a slightly more robust initial ship was required. This newly defined frigate (Testopt), differed from the example initial ship (Startship - Chapter 4) in the design displaced volume (increased from 2400m^3 to 3000m^3) and the prismatic coefficient (0.58 as opposed to 0.57). This produced some consequent modifications to the main particulars, see for comparison, Table 6.3. In order to perform the mapping analysis for tuning the constraints as discussed above, a "would be" optimum was estimated in a region considered to be possibly an optimal one, with (m) = 9.0 and B_{WL}/T = 3.5. This is shown in Table 6.3 as Testship with two versions that correspond to the two extreme flares. The total resistance of these two ships are shown in Figures 6.3, 6.4, 6.5 and 6.6. Note that for this modification it is expected that a higher optimal (minimum) resistance would result, compared with the example in Chapter 4. In all mappings C_P was fixed at 0.58 and L'_x at 0.52.

To illustrate the relations between the variation of flare and the dominance of the constraints two curves of B_{WL}/T vs $FLARE'_x$ were produced, one at (m) = 7.0 (Figure 6.7) and the other at (m) = 9.0 (Figure 6.8). It should be noted that since these relationships are independent of speed, the curves were made for metacentric height (GM) variation, instead of resistance, in order to separate GM from the

stability area constraints, to allow a clear comparison. As can be seen, greater flares favour stability or allow for smaller beams, as would be expected. It can also be seen from Figure 6.7 that the enclosed volume strongly restrains the function at lower flares with the stability taking over later on. It can also be noticed that for (m) = 7.0 (Figure 6.7) the stability areas are dominant over GM, while for (m) = 9.0 (Figure 6.8), at the highest flares, this tendency is isolated and GM (which is limited to 1.0m) becomes the dominant constraint, while the enclosed volume does not act any more.

Another interesting feature that helps to create a realistic difficulty to test the knowledge base's ability to control the optimization process can be seen from the structural strength constraint in Figure 6.8. This represents a two-dimensional cut of the highly constrained narrow five-dimensional corridors, mentioned in Chapter 4.

Finally, the two remaining design variables were plotted. Figures 6.9 and 6.10 represent the variation of the total resistance with respect to L'_x and C_P. For these figures B_{WL}/T = 3.5 and the initial points are for C_P = 0.58, L'_x = 0.52. Figure 6.9 is for (m) = 7.0 and Figure 6.10 is for (m) = 9.0. Figures 6.9a and 6.10a are for $FLARE_x$ = 0 deg. and Figures 6.9b and 6.10a are for $FLARE_x$ = 17.1 deg. It is interesting to note that all the boundaries represent infeasible ships, showing once more that the function is highly constrained in five dimensions.

6.5 Optimization example applying Knowledge Base 2

6.5.1 APPROX with two variables

Having set up the problem as explained in Sections 6.4.1 and 6.4.2, the next step was to run the initial ship testopt following the advice provided by Knowledge Base 2, see Tables 6.1 and 6.2 and Figure 6.2. The first rule asks the maps given as Figure 6.3 to be produced. Running APPROX, as advised by the rule base, the optimization was successful in 22 loops, as shown in Table 6.4, with a behaviour very similar to that described in Section 4.2.3. Then rule No.25 asks for a check on the optimum found. In the search for false optima it would not be necessary to carry out any extra mappings because there is one visible minimum and if this option is chosen the advice to n-dimensionalise and run APPROX again would be given by rule 30. If the user is unsure of the result and feels the resistance of R_T = 1050.80 kN may not lie in the mapping (Figure 6.3b), then he could select 'no visible minimum' and go to rule 26, which would ask for an amplification of the search area. This amplification would look like Figure 6.5c and he would be satisfied that the result so far is within reasonable engineering precision of the minimal result. The mapping also confirms that L_{WL}/D and the enclosed volume were the active constraints. The next choice is obviously to select the option of the lowest values of the objective function in 'only one region' (rule 27) that would then lead to the advice n-dimensionalisation (rule 30). Note that up to this moment the resistance has been improved by 40%.

6.5.2 APPROX with five variables - first run

Next, rule No.1 is modified and advises APPROX to be run starting from the point rule 30 suggested be preserved for the n-dimensional run. In this next run APPROX fails to converge after 41 loops (see Table 6.4). The reason for this non-convergence is the failure to build a hyper-plane due to the inconsistent hull-forms that arise (see Section 4.2.4). When consulting the advisor the user is asked whether, in his judgement, the function is suitable for further linearisation (rule 6). Explanation on how to decide this is provided, pointing out that the function could be highly non-linear or that the design process could have been driven into an infeasible region from which it cannot get out. The advisor goes on to suggest that the output files be studied and shows how to judge the behaviour from the results. An experienced naval architect would most probably conclude two things at this point: (1) that the function is either highly non-linear or highly constrained at the point where it failed to converge or (2) that the results up to this point could be considered good enough from the engineering point of view because the conflicting constraints, as seen in Tables 6.2 and 6.4 are close to their limits. The total resistance of 858.35 kN achieved at this point represents a further improvement of 22%.

To decide which of these conditions apply it can be seen that, in fact, the function does not present a behaviour that would make it non-linearisable, but comparing Figure 6.8 with Figure 6.7, at a length to displacement ratio of 9.0 it does indeed become heavily constrained. From this, one would conclude that the function cannot be linearised and the advisor would then go to rule No.6 and suggest using the direct

search method SEEK, with a brief explanation on the method and the penalty functions. Soon after, rule No.2 would give the details of running SEEK with OPTIM 1.

However, following the second line of reasoning, one could decide to end the optimization process and perhaps perform some manual runs around the last point to finalise the results. Since there is no such advice for this second conclusion, this is a shortfall detected in Knowledge Base 2 that should be corrected. This will be discussed later in Section 6.6. A less experienced naval architect (or an optimistic one) might consider the function suitable for further linearisation, since the fact that there is no advice to check if it was time to stop, brings some insecurity to the user. Nevertheless, it must be said that rule No.1 advises the user to produce mappings of cross-sections using the remaining variables, according to the mapping strategy of Section 3.5. Because the knowledge base must be general, for any sort of function, this advice must be vague and therefore some of it may be misunderstood, or the maps misinterpreted.

Another opposing aspect of this is that, surprisingly, the results obtained so far are better than the initial mappings had suggested they would be. They have also been achieved more quickly and occurred half way through the flare range tested suggesting there was an unpredicted 'bend' to better results in the hyper-space which was not seen when only the extremities were mapped. This highlights the idea that the mapping strategy is not foolproof and that optimization really has a contribution to make. This is further discussed in Section 6.6.

6.5.3 APPROX at a closer starting point combined with step size tuning

To continue testing Knowledge Base 2, it was decided to select the option of trying to linearise the function further. Therefore, rule No.7 suggested a check on the building of the hyper-plane and also to try a closer starting point (rule No.1 modified), using the result of the previous run. This run of APPROX failed to converge after 6 loops due to inconsistent hull forms (see Table 6.4). Although the method could build a hyper-plane, it could not 'jump' to a viable ship. The next advice then was, quite properly, to tune the size of the steps (rule No.8), while rule No.1, again modified, suggested a value for the step size reduction which was adopted. This optimization run of APPROX declared failure after 13 loops (see again Table 6.4), but the smaller size of the steps brought a slight improvement in the resistance (of 0.5%), increasing (m), the flare and the position of maximum beam and decreasing L_{WL}/T and C_P. In this case the area under the stability curve from 0 to 30 degrees was taken to its limit. Although negligible, this improvement pointed the search in the right direction and tried to explore an extra corridor suggesting that this rule might be necessary for other cases in a more general context.

6.5.4 Changing to SEEK

Having failed again, the advisor suggested (rule No.9) that it was unlikely APPROX would work further, but that some localised runs of short RANDOM searches could be used to try to find a better point. This rule was found necessary in the studies of Knowledge Base 1 in order to try to guarantee that every possible benefit of using APPROX was taken.

If any improvement is found a further run of APPROX from the point achieved is tried (rule 10), until improvements were exhausted and then rule No.11 would finally suggest a move to SEEK. From the engineering precision point of view the example test ended up showing that this region offered no possible improvements and therefore it was decided to move straight to rule 11. Rule 11, as rule 6, leads to rule 2, i.e., the set-up to run SEEK with OPTIM 1.

At this stage one may have realised that there was not much point in further trying to reduce resistance at all. However, for the sake of testing the knowledge base the process was followed through. The rationalisation here was to neglect differences in engineering precision and try to achieve smaller results of the objective function in the hope of finding some of these in the highly flared extremity of the hyper-space, trying to explore most of the objective function. This was done in the knowledge that the mathematical ship design model could be too simple for the knowledge base and optimizers' capacities, and that the objective function in a more complex example, could contain several parameters where these differences would be significant.

6.5.5 SEEK with OPTIM 1

SEEK with OPTIM 1 was run from the results of run number 4 (see again Table 6.4). The behaviour of the optimization method was very much affected by the new highly constrained example. When inconsistent hull-forms arose the search was successfully driven back to path, similarly to the behaviour in Section 4.2.4., but after 611 loops it declared a hang up with all random shots used, which leads to rule

No.12, i.e., change to SEEK with OPTIM 2. The advisor explains that SEEK with OPTIM 1 cannot do better since it has tried all its possibilities. And indeed no better result was found in the output file, although the search had been driven into the highly flared region, finding good results, but none better than the previous ones. The method coped well with inconsistent hull forms and was driven successfully back to the original area. A look at the limits tested suggested it covered a large part of the feasible area of the function, but the problem here seemed to be that there was nothing better to find. The most active constraints were the stability and L_{WL}/D ratio.

6.5.6 SEEK with OPTIM 2

Following the instructions of rule No.12, SEEK with OPTIM 2 was run, according to the details given by rule No.3. The initial point used was one of the best results given by run No.5. This method declared failure to converge after 1217 loops. Nevertheless before failing it produced some interesting, low values of the objective function by systematically trying to lower C_P beyond its lower limit. Another feature noticed was that it tried to produce a slightly shorter ship getting close to the enclosed volume limit. It is a characteristic of softer penalty functions to violate constraints which sometimes may suggest improvements may be gained by relaxing these constraints, as seen in Section 4.2.4. The natural outcome of a failure after trying this is to declare there is no feasible solution. Knowledge Base 2 was programmed to take this into account and suggest a relaxation of constraints if at all possible - rules 16 and 22 (as explained in Section 6.3). The case of run No.6 is in fact a case of non-convergence, but the attempt to

decrease C_P, thus violating this constraint, was very frequent. Therefore it was decided to repeat the same run relaxing the C_P lower limit to 0.53. This was run No.7 and the process declared success after 323 loops, giving a result, as would be expected, with a very slight improvement. What had occurred was that the new boundary gave flexibility to allow the method's strategy to converge, finding a solution in the original region, not within the newly released area. If the user then selected 'success' from the knowledge base menu, this would have ended the advisory session, through rules No. 31 and 32.

6.5.7 Random Search

Finally, a random search minimising the shrinkage factor was conducted, supposing that the user did not relax the constraints for SEEK with OPTIM 2, thus following rules No.21 and 4. The shrinkage factor was reduced setting the system to take 10,000 design samples, keeping the best 16 out of 800 combinations of the five variables, as opposed to the example of Section 4.2.4. - RANDOM, where 200 samples were taken, allowing for 5000 combinations of variables. This was found necessary by trial, following the advice of rule No.4; the function was so constrained that it did not get the lowest results using 400 samples. The result is given was :-

R_T	= 866.20 kN	(m)	= 8.903
B_{WL}/T	= 3.52	C_P	= 0.5536
L'_x	= 0.507	$FLARE'_x$	= 3.690
		($FLARE_x$	= 6.479 deg.)
L_{WL}/D	= 13.67	ENC.VOL	= 17293

and positive stability.

To give an order of magnitude of the time consumption, this run took 30,135 loops to finish.

6.5.8 Optimization with five variables without using the Knowledge Base

For the sake of comparison, two five-dimensional optimizations were run from the initial example ship testopt, as if the knowledge base or the mapping strategy did not exist. The first was APPROX and the second SEEK with OPTIM 1, both using default features.

The APPROX run declared non-convergence after 41 loops. The last feasible results before failure where those of loop 26:

R_T	= 871.72 kN	(m)	= 9.014
B_{WL}/T	= 3.56	C_P	= 0.5651
L'_x	= 0.497	$FLARE'_x$	= 3.1500
		($FLARE_x$	= 5.593 deg.)

The SEEK with OPTIM 1 method declared hang up, with all shots used, after 801 loops. The last feasible results before failure found were on loop 794:

R_T	= 861.33 kN	(m)	= 9.000
B_{WL}/T	= 3.56	C_P	= 0.5566
L'_x	= 0.5877	$FLARE'_x$	= 2.519
		($FLARE_x$	= 4.479 deg.)

In both cases L_{WL}/D was at the limit, the enclosed volume was small but not minimised and there were slightly better results with small infeasibilities in the output files, which would not be found without advice or knowledge that they could be acceptable.

These results would seem to deny the need for a knowledge base if the results considered were just required to engineering precision. This is in fact due to the difficulty of trying to produce an example that would allow complete demonstration of the knowledge base. If on the one hand the example made it difficult for the optimizers to achieve what was thought to be the optimum, on the other it could not be predicted that there would have been better results and that these would be found before the function got so constrained that the optimizers would no longer work. In fact, as a coincidence, the optimum seemed to lie in the proximity of the region where all the methods fail. When helped by the knowledge base's advices to move further, no better points are found.

6.6 Conclusions and Improvements

At first sight there may seem to be little advantage in using the knowledge base to control the optimization process. Nevertheless it should be noted that: (1) the problems with the example used were in the precision of the objective function - these differences could be significant, and would have a different meaning, in a more complex function; (2) running optimization alone, since it declares failure and as there would not be any mapping available, the user would not know if the results would be useful or how good would they be. Since the major

criticisms of optimization are lack of credibility, without the knowledge base or some other expertise there is no way the user can interpret the results if the method declares failure and there is no possibility of knowing if the result is a false optimum without mapping, even if it is successful.

Another point is time consumption. Supposing the mistakes found in Knowledge Base 2 are corrected, this example case would have ended giving a reasonable feeling of an optimum being achieved after run No.2, thus adding up to 63 loops, with an improvement of 71% in the resistance, while the direct runs of APPROX would have had to be confirmed either by a RANDOM search or by SEEK with OPTIM 1, which declared failure after 811 loops.

As mentioned before, the fact of the process finding an optimum that was unpredicted shows that a mapping strategy cannot be considered foolproof and that optimization has in fact a contribution to make if well guided. For these reasons and the results achieved it seems that such a knowledge base system can bring credibility and time saving to the optimization process, but it still needs corrections and improvement.

The improvements to the knowledge base that would be immediately required are :

(1) A better explanation of the mapping strategy at the beginning of the process should be provided, in order to make sure the user would start optimizing having at least figures of the type 6.7 to 6.10, and an understanding of their meaning. This is stated in

rule No.1, but this is still too vague. Having these figures and analysing the function as advised by the knowledge base, the user would feel confident of realising when he had reached the end of an optimization.

(2) An advice should be included to judge results at each stage to see if they are good enough to stop optimization.

(3) It would be useful to include advice for extra two-dimensional mappings after an n-dimensional run around the region where the optimizer cannot move further. These could be very efficient using one or two maps with two or three of the dominant variables. This would show the behaviour of the function in this region, such as its smoothness and also any local variations in the objective function.

(4) The knowledge base already suggests that even having failed there could be useful designs in the output file of an optimization run. It still does not explain that these results might be the best that are possible and this should be included.

(5) It would also be useful to have optimization used to "drive" the solution to feasible "best" areas and from these produce mappings from which the user would decide which ship to select from the bounded areas. To achieve this, suitable advice should be added to the knowledge base for use at the end of a consultation. This idea was tried using the solution found after run No. 2. The procedure used was as follows: (1) Produce a map based on the last

best point found and its surroundings for the two most dominant variables, fixing the rest (Figure 6.11). (2) From this figure extract a ship with sensible rounded up dimensions, i.e., the length was rounded to L_{WL} = 130.00 m and the beam to B_{WL} = 13.12m, see Table 6.3. (3) From this point produce relevant mappings for combinations of the remaining variables (Figures 6.12 and 6.13). Now all the objective and constrictive goals are under control and the user can choose his configuration within this "best found" area.

Another future development concerns the integration of the system as a whole. Controlled by the user, the knowledge base would call the optimization routines and run them. In a fully integrated system the knowledge base would also cater for the design synthesis, as mentioned in Chapter 5 and optimization control would then be just a small part of of the whole task.

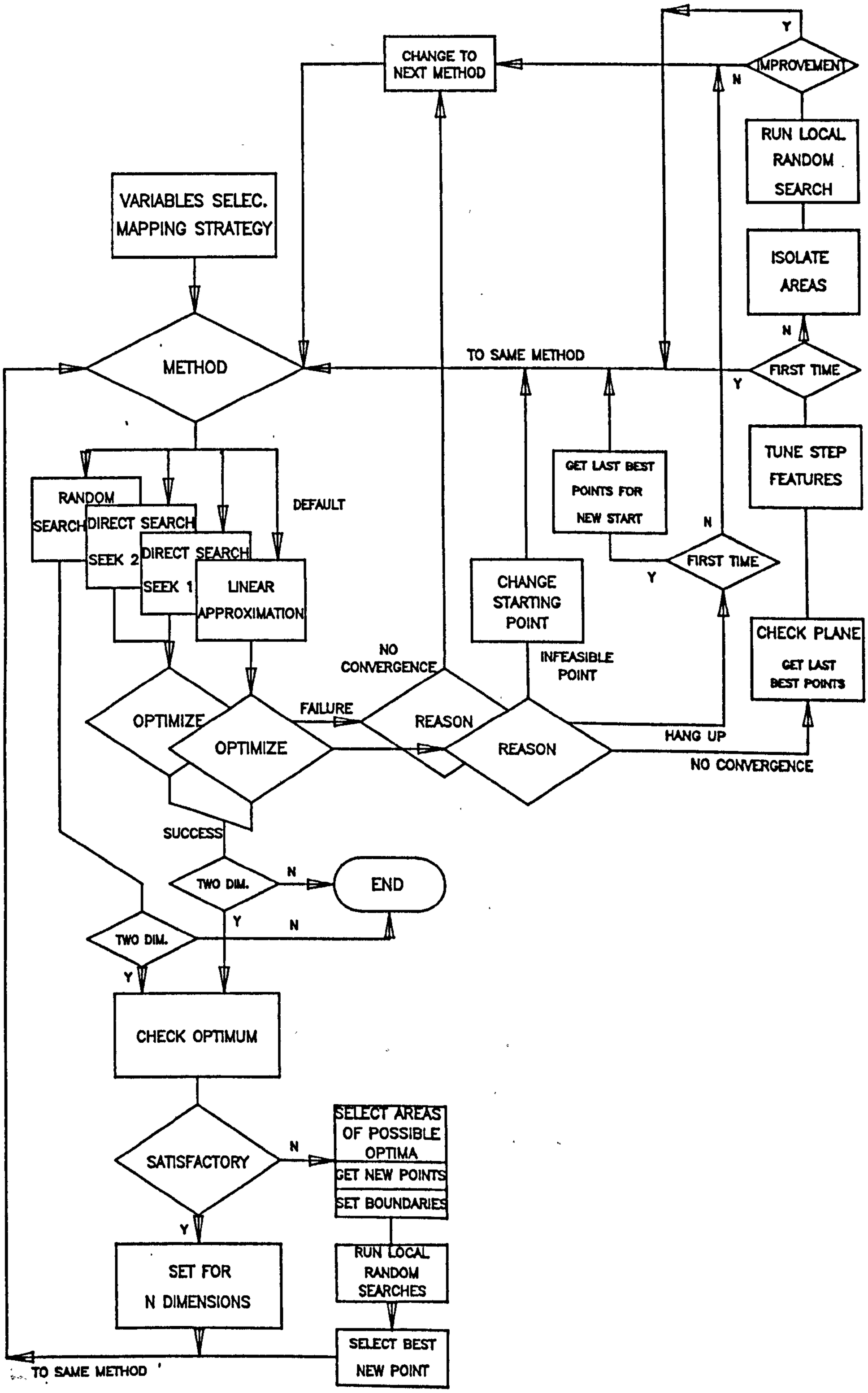

Figure 6.1 – Knowledge Base 1 (flow–chart)

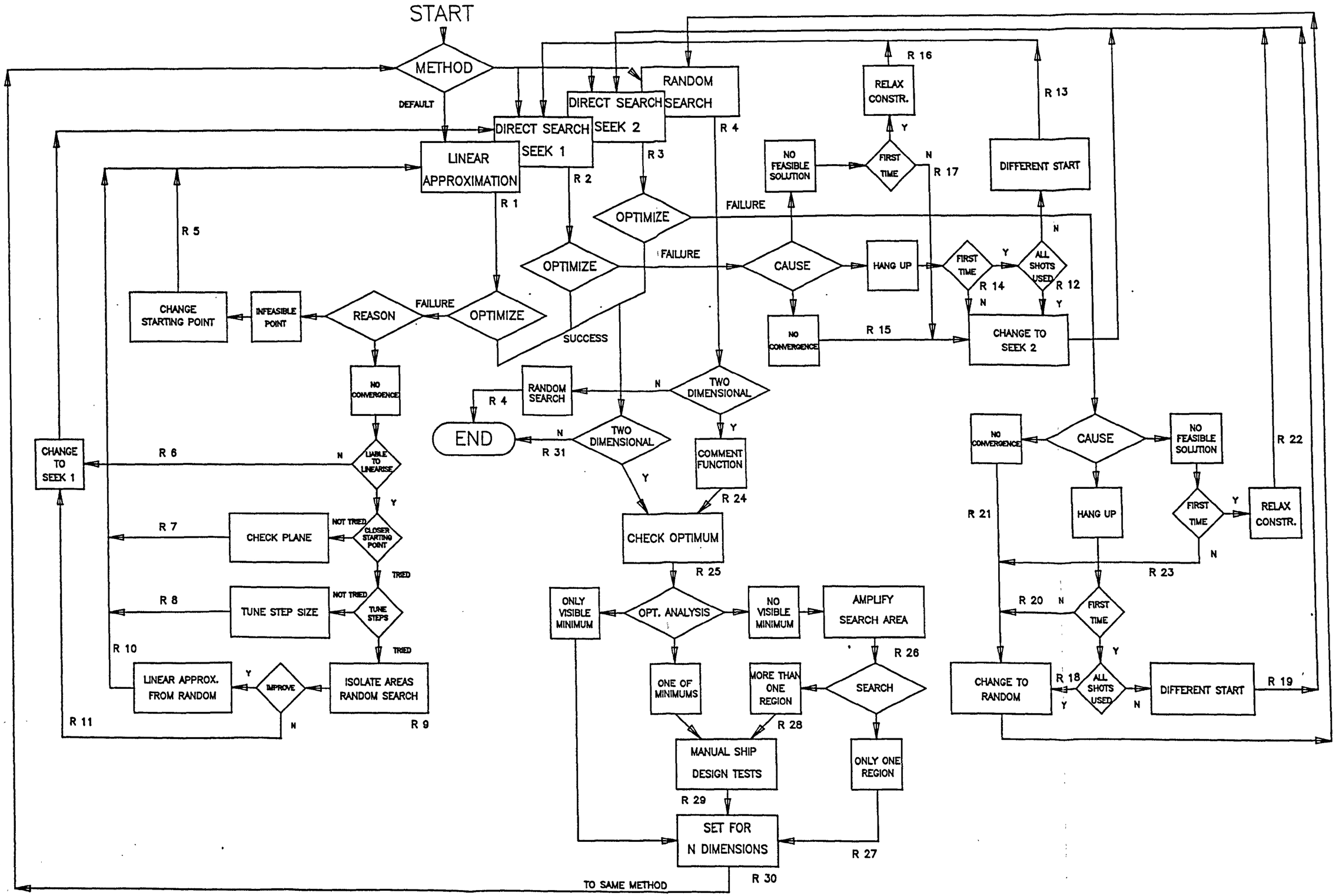

Figure 6.2 – Knowledge Base 2 (flow–chart)

* R i, (i=1, 31) denotes rule number in the knowledge base

ⓜ	7.0	C_P	0.58
B_{WL}/T	4.0	∇	3000 m^3
C_B	0.47	D/T	2.5

Table 6.1 - Initial input data.

Optimization	with 2 variables	with 5 variables
Objective Function	Total Resistance R_T (KN)	
Constraint Vector and Limits	Initial GM ≥ 1.0 m. Area below GZ curve from 0 to 40° ≥ 0.27 m rad. Area below GZ curve from 30 to 40° ≥ 0.09 m rad. Area below GZ curve from 0 to 30° ≥ 0.165 m rad. Angle of Maximum GZ ≥ 30°. Maximum GZ ≥ 0.2 m. $L_{WL}/D \leq 14.0$ $17000m^3 \leq \nabla_{ENC}$ $0.0 \leq L_{WL} \leq 1000.0$ m $0.0 \leq B_{WL} \leq 1000.0$ m $0.0 \leq T \leq 1000.0$ m $0.0 \leq D \leq 1000.0$ m $0.4 \leq C_X \leq 1.0$	
Trial Vector and Limits	$5.0 \leq$ ⓜ ≤ 12.0 $1.0 \leq B_{WL}/T \leq 6.0$ $C_P = 0.58$ $FLARE'_X = 0.0$ $L'_X = 0.52$	$5.0 \leq$ ⓜ ≤ 12.0 $1.0 \leq B_{WL}/T \leq 6.0$ $0.55 \leq C_P \leq 0.60$ $0.0 \leq FLARE'_X \leq 10.5$ $0.14 \leq L'_X \leq 0.60$
Fixed Parameters	$\nabla = 3000m^3$ $C_B = 0.47$ $D/T = 2.5$	

Table 6.2 - Optimization Conditions (Summary).

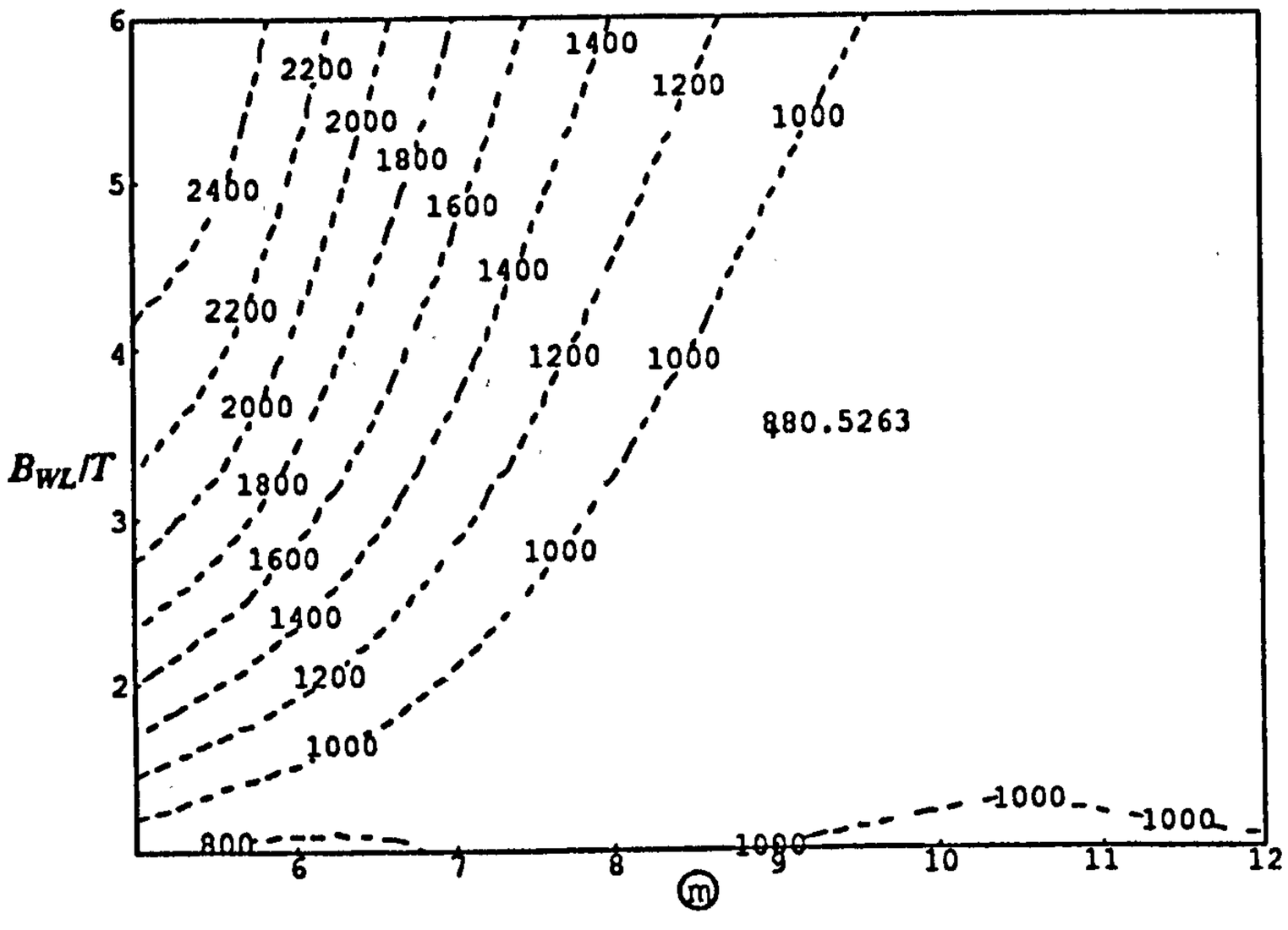

Figure 6.3a - Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T for testship, unflared, unconstrained.

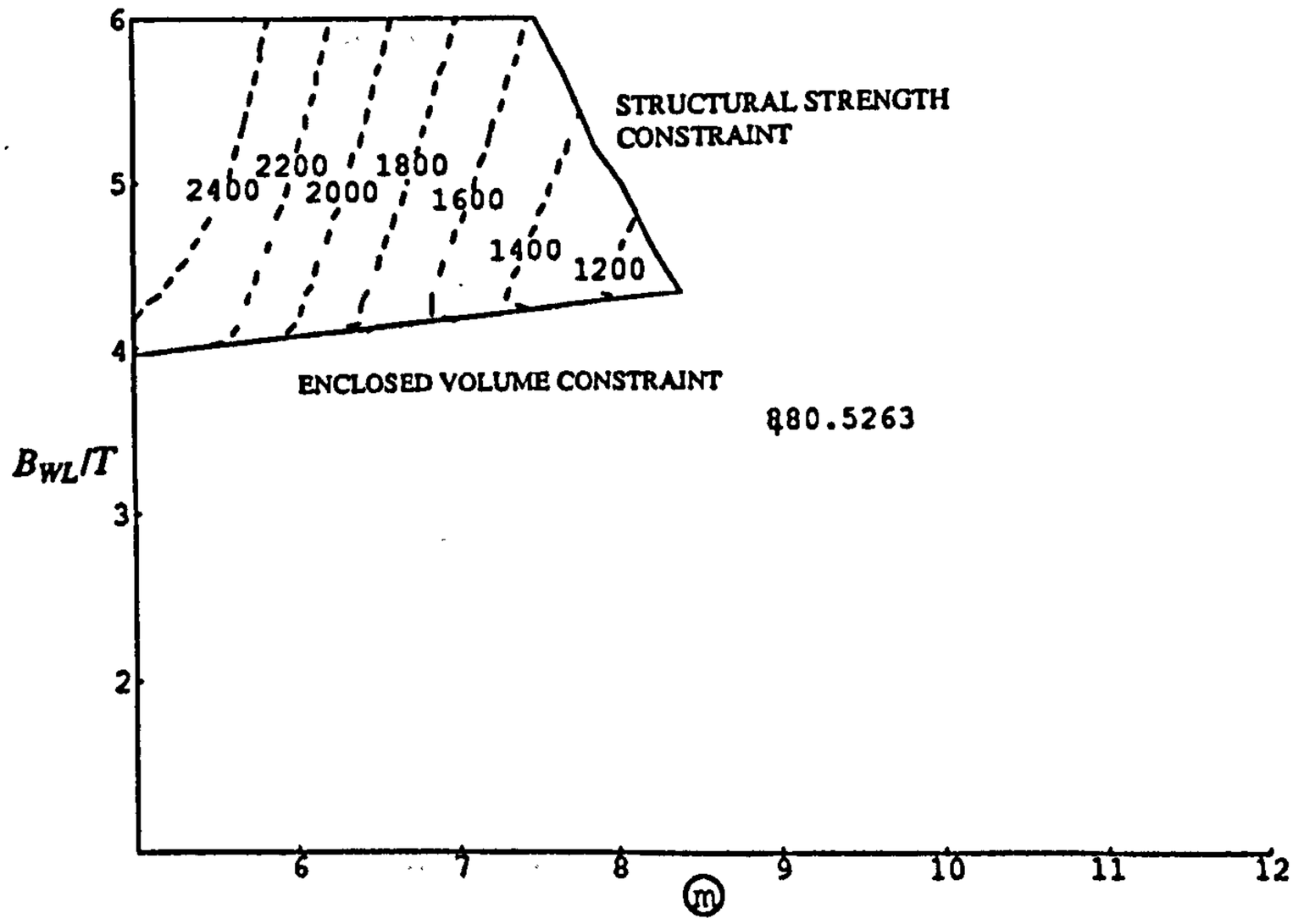

Figure 6.3b - Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T for testship, unflared, constrained.

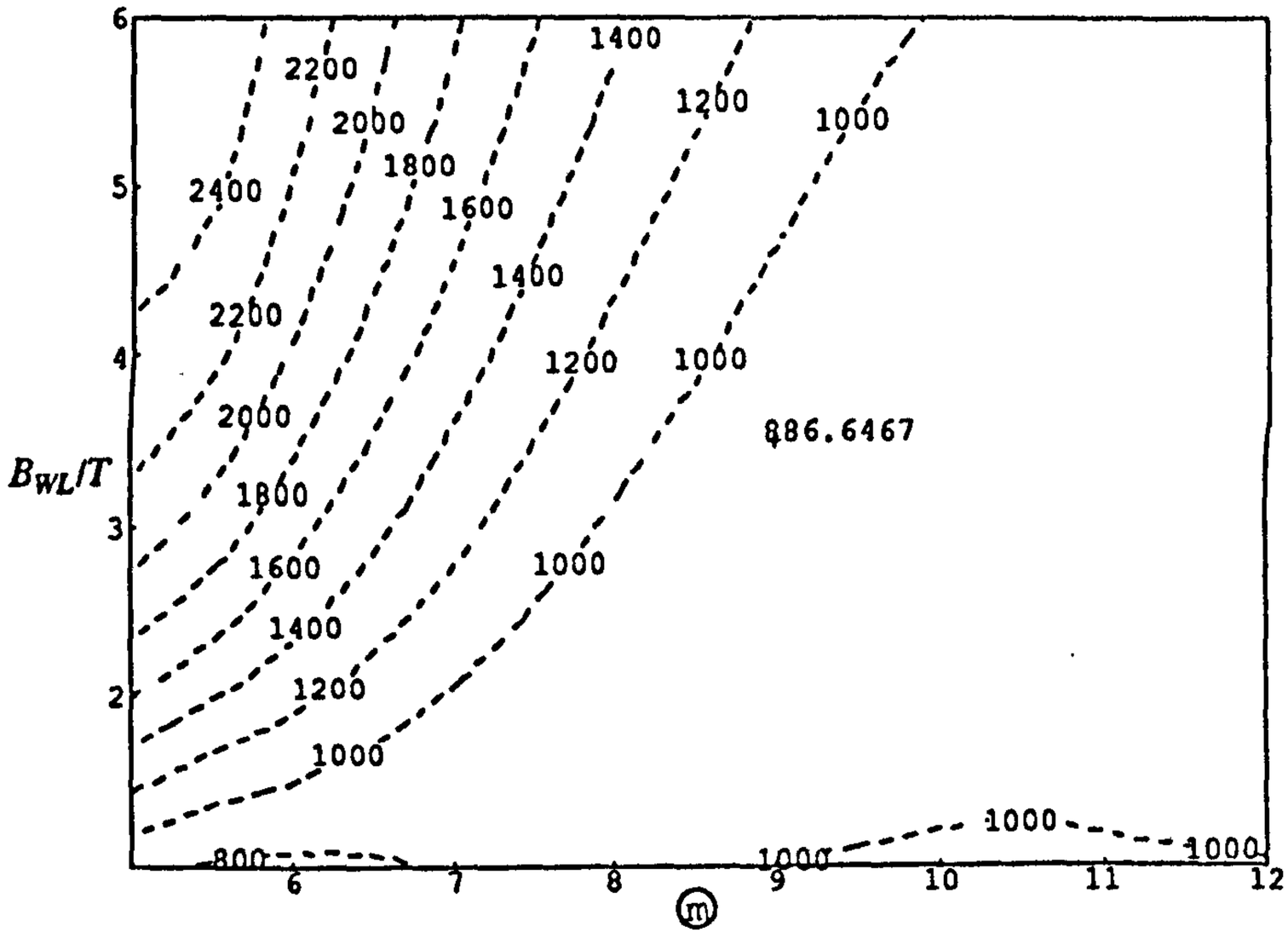

Figure 6.4a - Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, (m) versus waterline beam to draught ratio, B_{WL}/T for testship, flared, unconstrained.

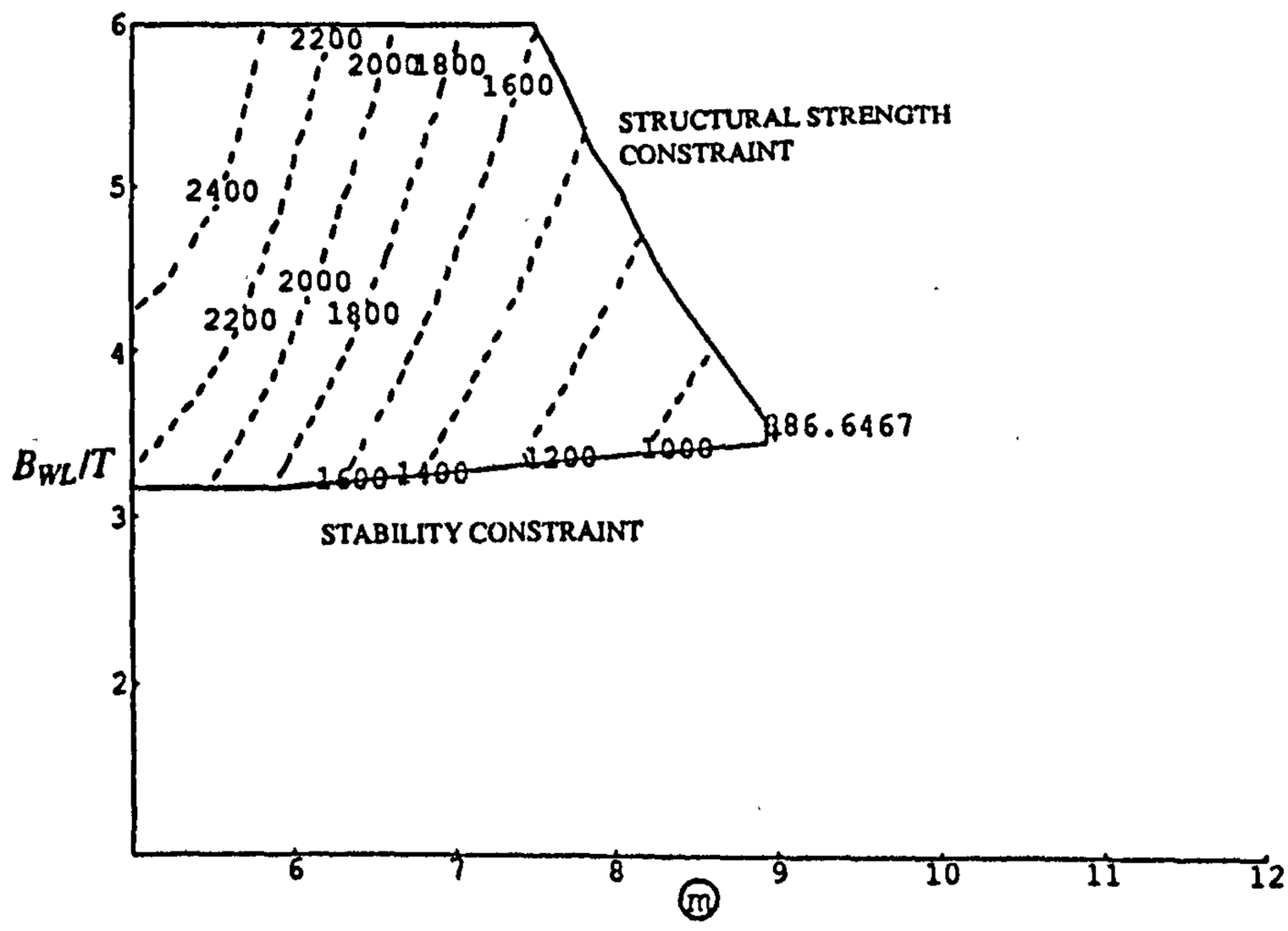

Figure 6.4b - Contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, (m) versus waterline beam to draught ratio, B_{WL}/T for testship, flared, constrained.

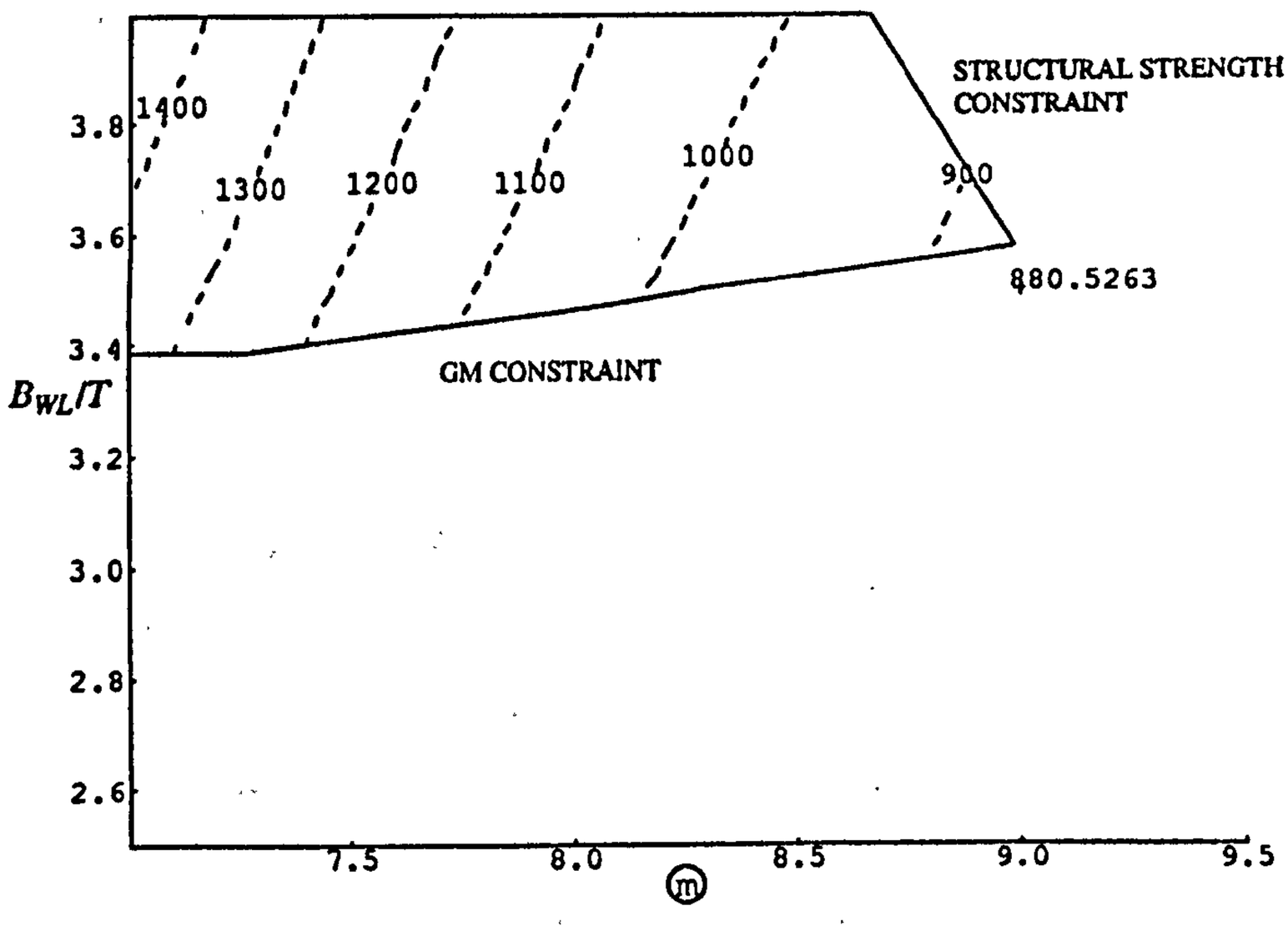

Figure 6.5a - Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T constrained only by structural strength and GM, unflared.

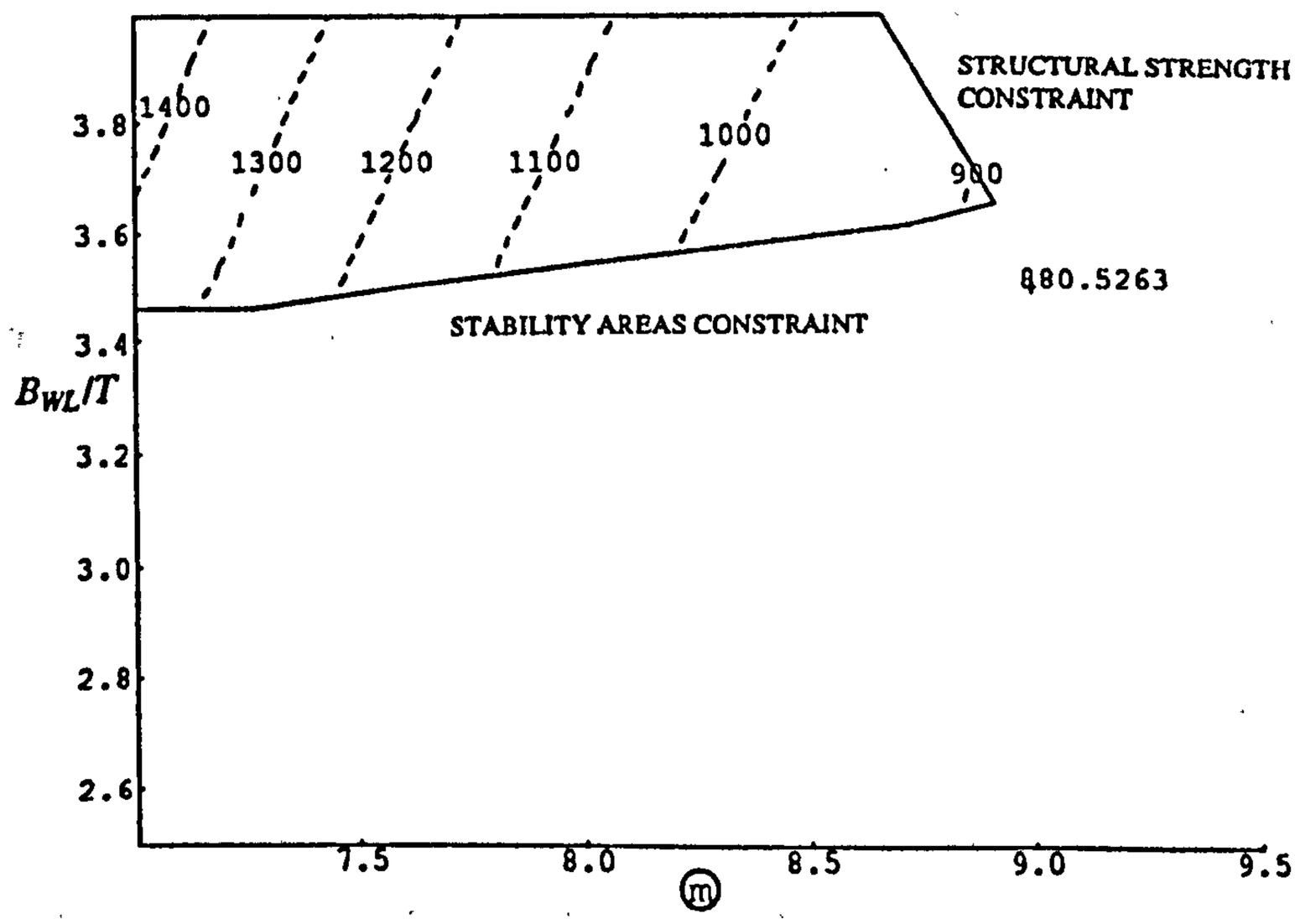

Figure 6.5b - Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T constrained only by structural strength and the stability curve areas, unflared.

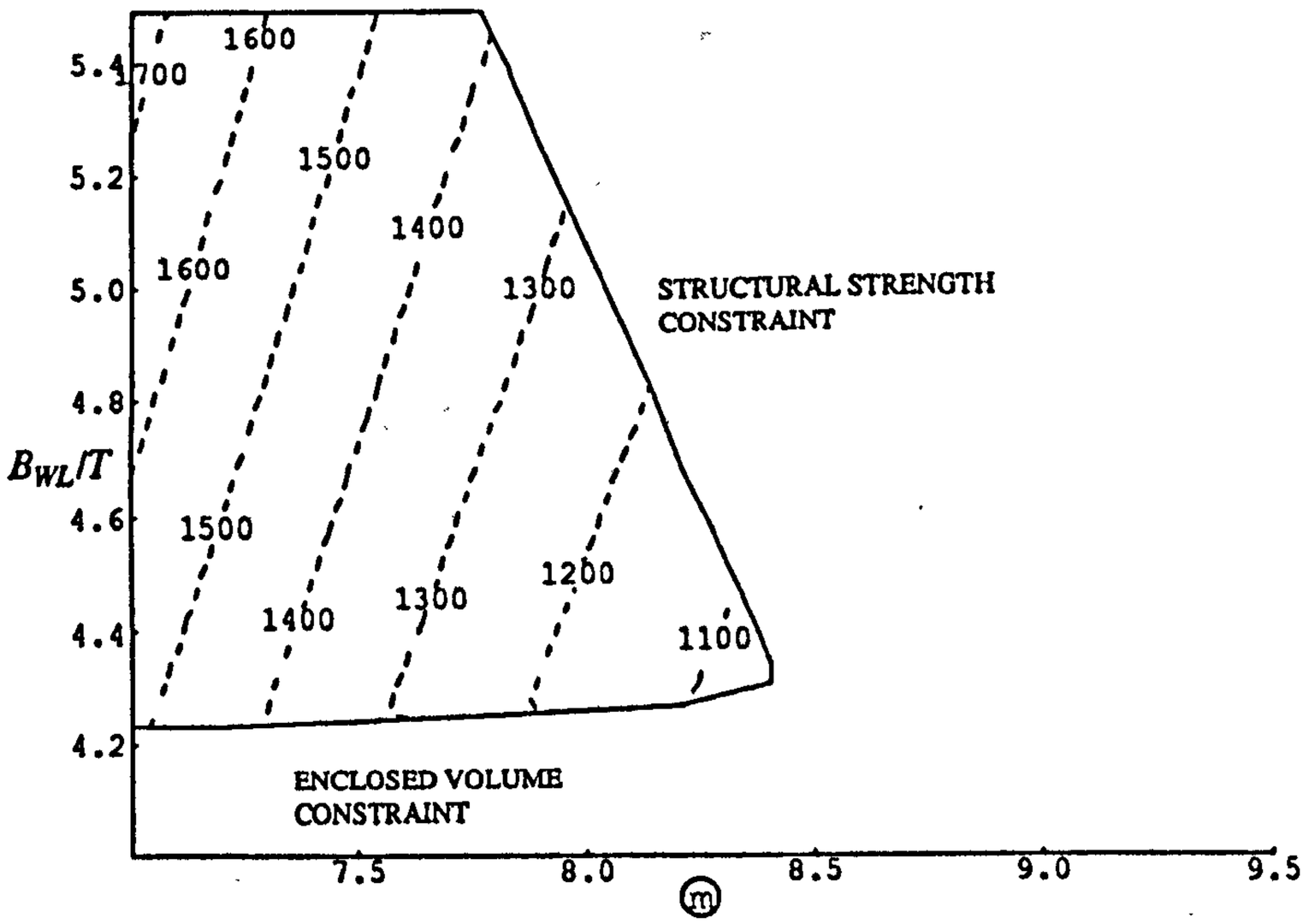

Figure 6.5c - Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T constrained only by structural strength and the enclosed volume, unflared.

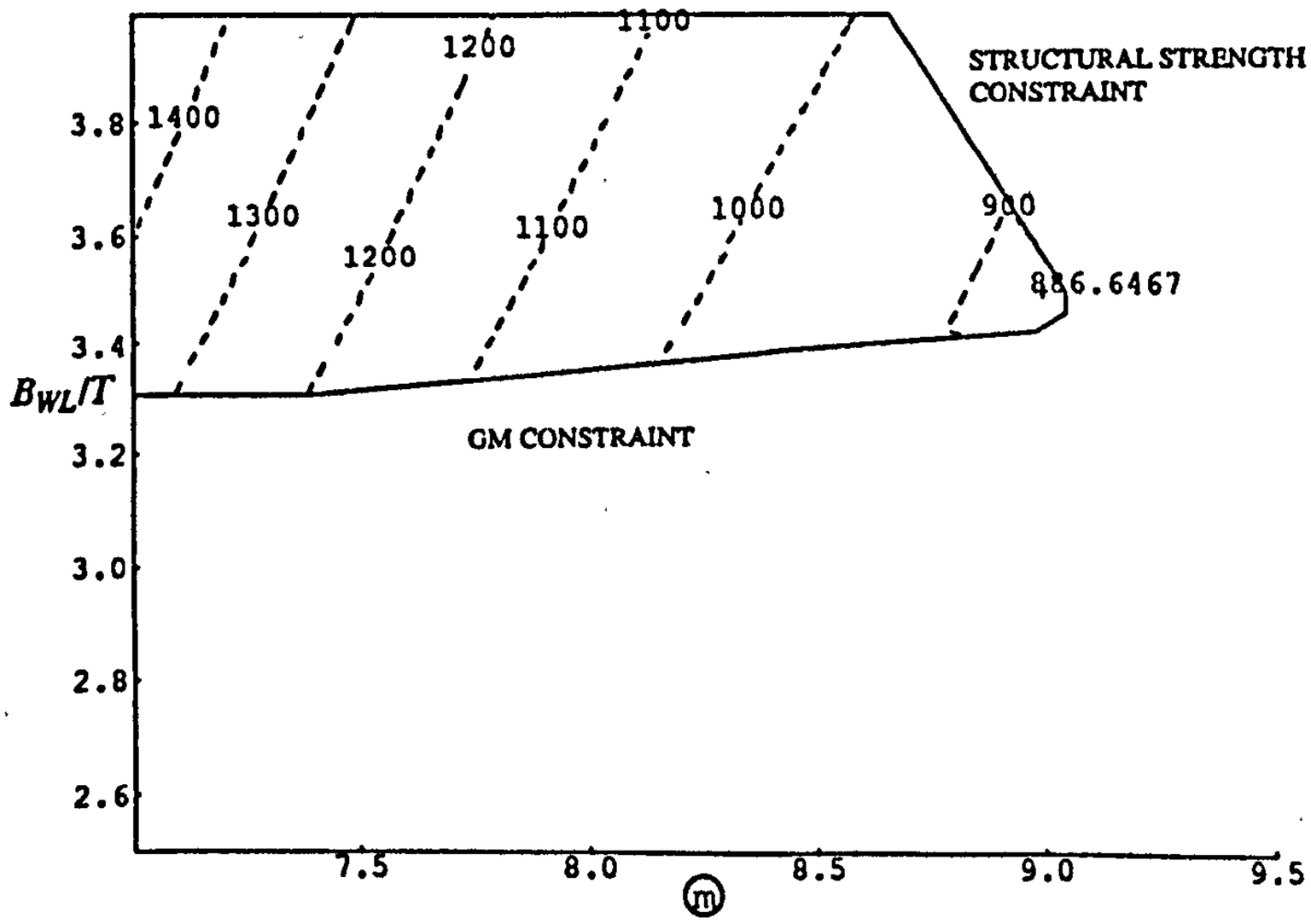

Figure 6.6a - Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T constrained only by structural strength and GM, flared.

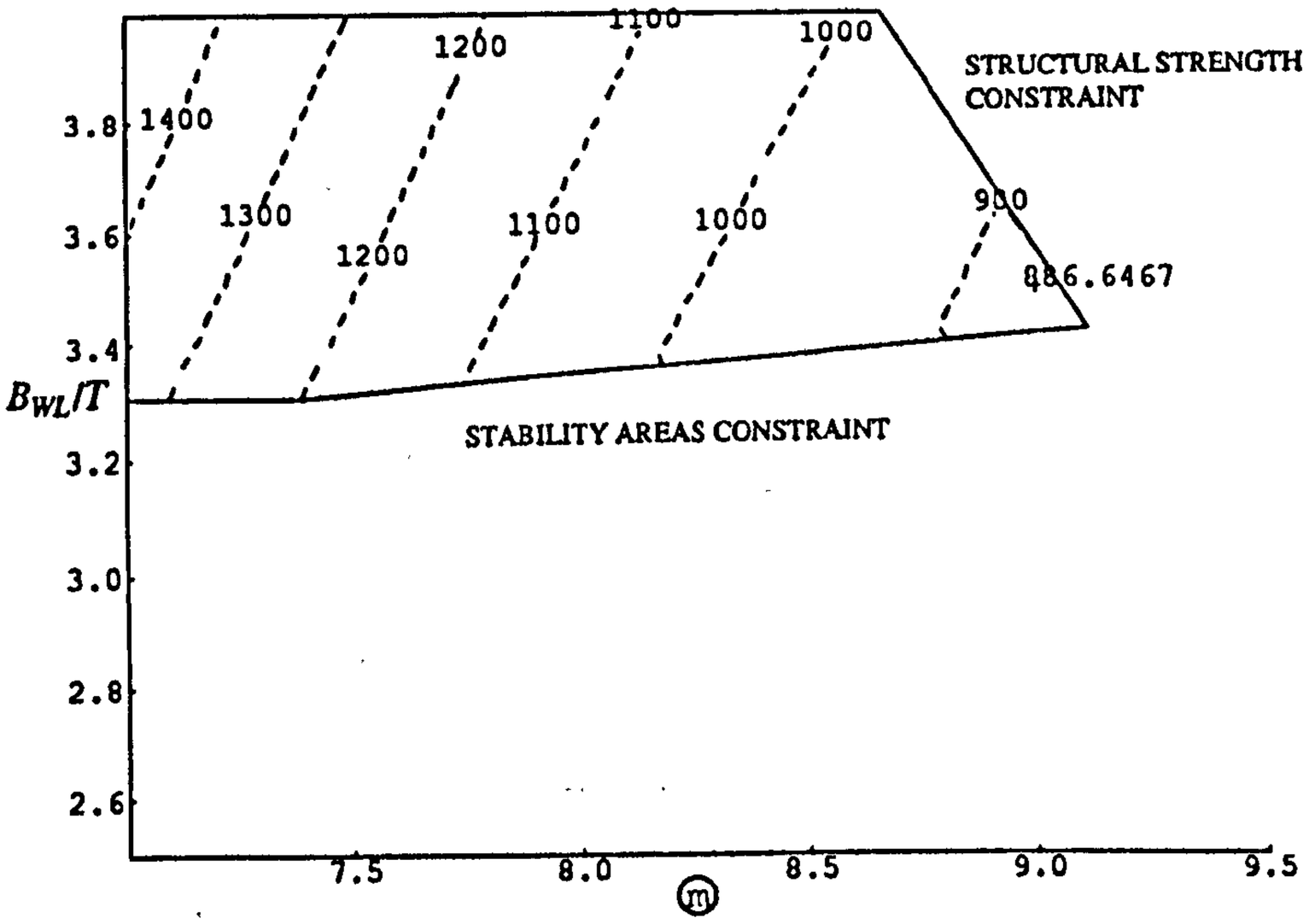

Figure 6.6b - Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, (m) versus waterline beam to draught ratio, B_{WL}/T constrained only by structural strength and the stability curve areas, flared.

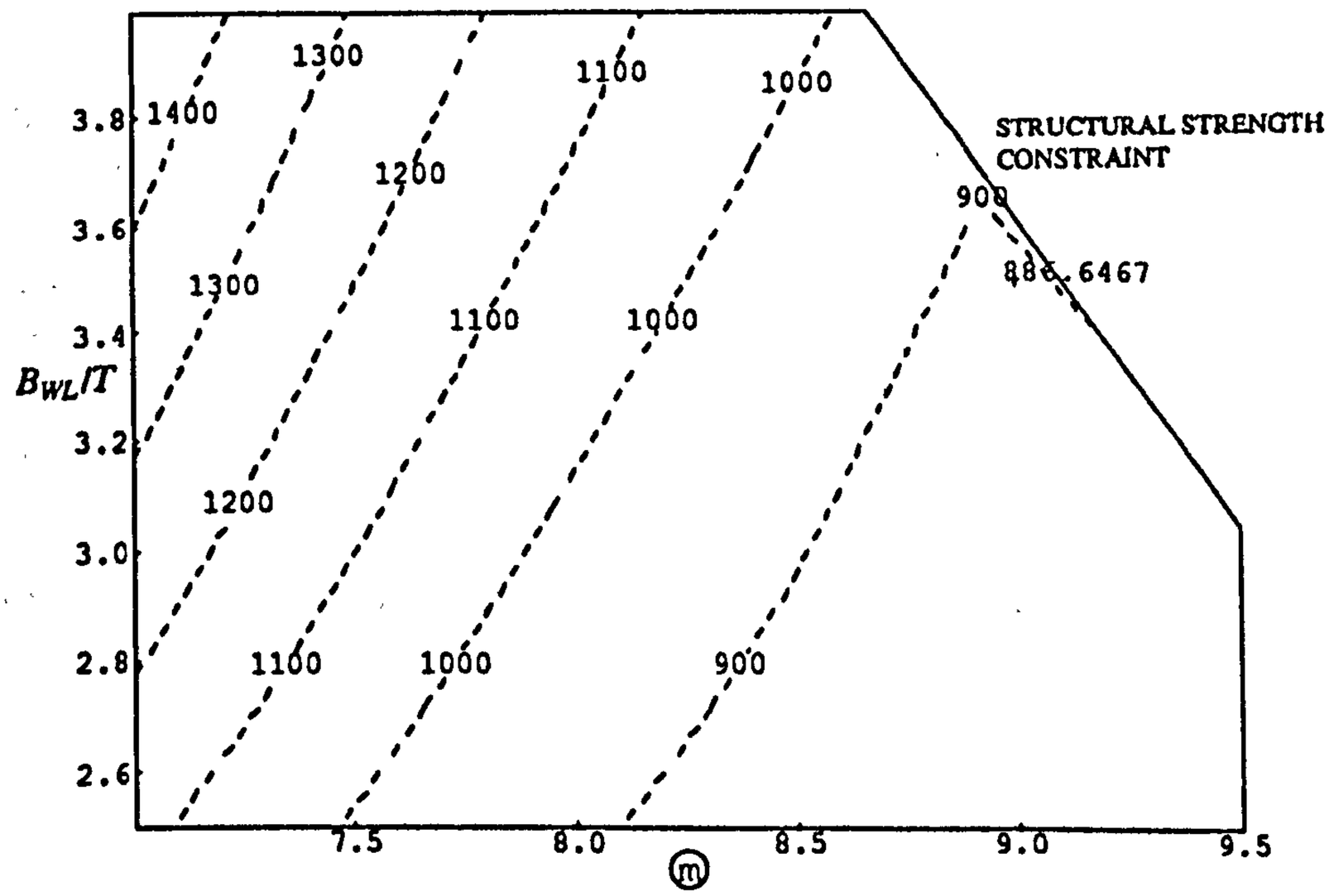

Figure 6.6c - Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, (m) versus waterline beam to draught ratio, B_{WL}/T constrained only by structural strength and the enclosed volume, flared.

CONFIGURATION	Startship	Testopt	Testship		Final Design
			flared	unflared	rounded up conf.
R_T (KN)	1505.80	1472.77	886.65	880.53	858.55
ⓜ	7.00	7.00	9.00	9.00	9.014
B_{WL}/T	4.00	4.00	3.50	3.50	3.506
C_P	0.57	0.58	0.58	0.58	0.55
$FLARE'_X$	0.0	0.0	10.0	0.0	5.25
L'_X	0.52	0.52	0.52	0.52	0.566
L_{WL} (m)	93.80	100.96	129.80	129.80	130.00
B_{WL} (m)	14.77	15.90	13.12	13.12	13.12
T (m)	3.693	3.976	3.748	3.748	3.742
D (m)	9.23	9.94	9.37	9.37	9.36
C_X	0.825	0.810	0.810	0.810	0.847
C_{WP}	0.722	0.722	0.722	0.722	0.720
B_{OA} (m)	14.77	15.90	16.59	13.12	14.93
BT_{WL}/B_{WL}	0.45	0.45	0.45	0.45	0.45
BT_{OA}/B_{OA}	0.95	0.95	0.95	0.95	0.95
$FLARE_X$ (°)	0.0	0.0	17.2	0.0	9.15
$FLARE_A$ (°)	35.0	35.0	31.5	31.5	31.5
ROF (°)	2.5	2.5	2.9	2.9	2.9
L_{MID} (m)	1.80	1.94	2.49	2.49	2.50
L_{KEEL} (m)	52.0	56.0	71.98	71.98	72.1
i_E (°)	13.0	13.0	8.4	8.4	8.4
$i_{E\ UD}$ (°)	25.0	25.0	16.7	16.7	16.6
$RAKE_F$ (°)	33.0	33.0	41.5	41.5	41.6
$RAKE_A$ (°)	0.0	0.0	0.0	0.0	0.0
Initial GM (m)	2.01	2.30	1.14	0.89*	1.03
Area 0-40° (m rad)	0.52	0.54	0.38	0.25*	0.32
Area 30-40° (m rad)	0.23	0.23	0.19	0.12	0.16
Area 0-30° (m rad)	0.29	0.310	0.188	0.134*	0.165
Angle Max GZ (°)	49	49	52	54	52
Max GZ (m)	1.57	1.62	1.51	0.96	1.28
L_{WL}/D	10.15	10.15	13.85	13.85	13.90
$\nabla_{ENC}(m^3)$	14065	16808	18731	16371	17592
Δ_D (t)	3274	3594	3845	3845	3847

Table 6.3 - Comparisons and details of the configurations with the final design.
* insufficient stability

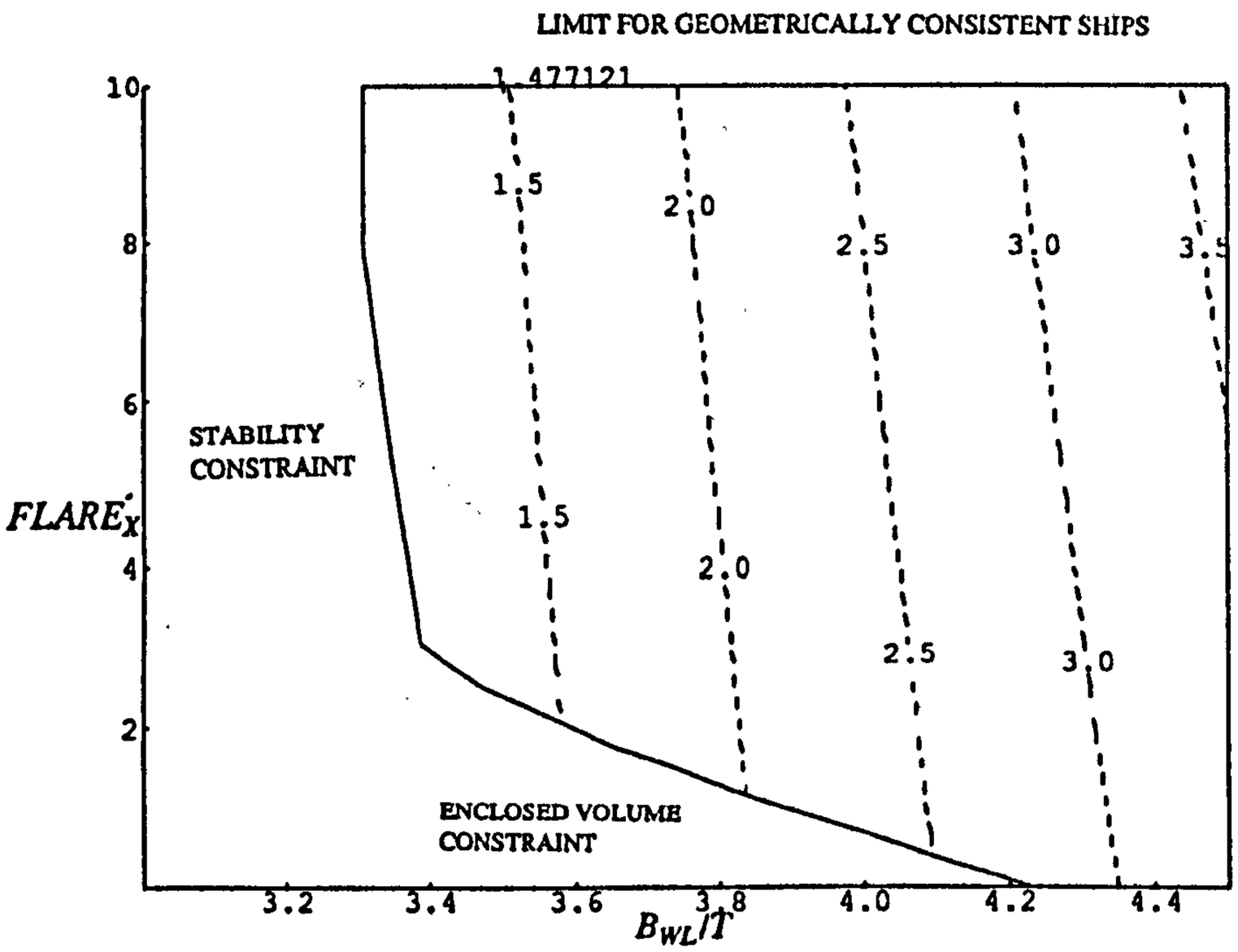

Figure 6.7 - Contour map of the metacentric height, GM (m) for waterline flare at maximum beam, $FLARE_X^{\cdot}$ versus waterline beam to draught ratio, B_{WL}/T, at ⓜ=7.0.

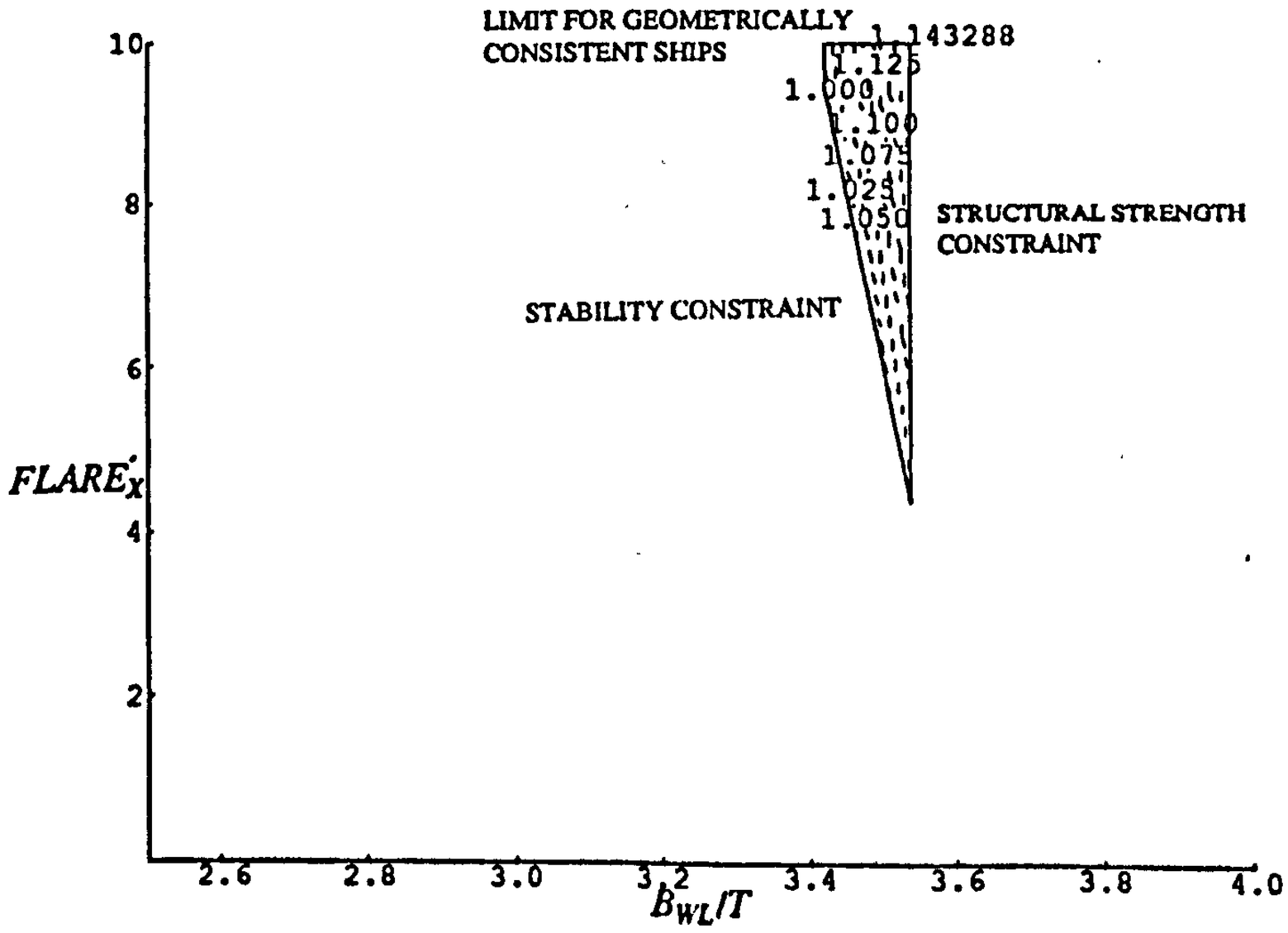

Figure 6.8 - Contour map of the metacentric height, GM (m) for waterline flare at maximum beam, $FLARE_X^{\cdot}$ versus waterline beam to draught ratio, B_{WL}/T, at ⓜ=9.0.

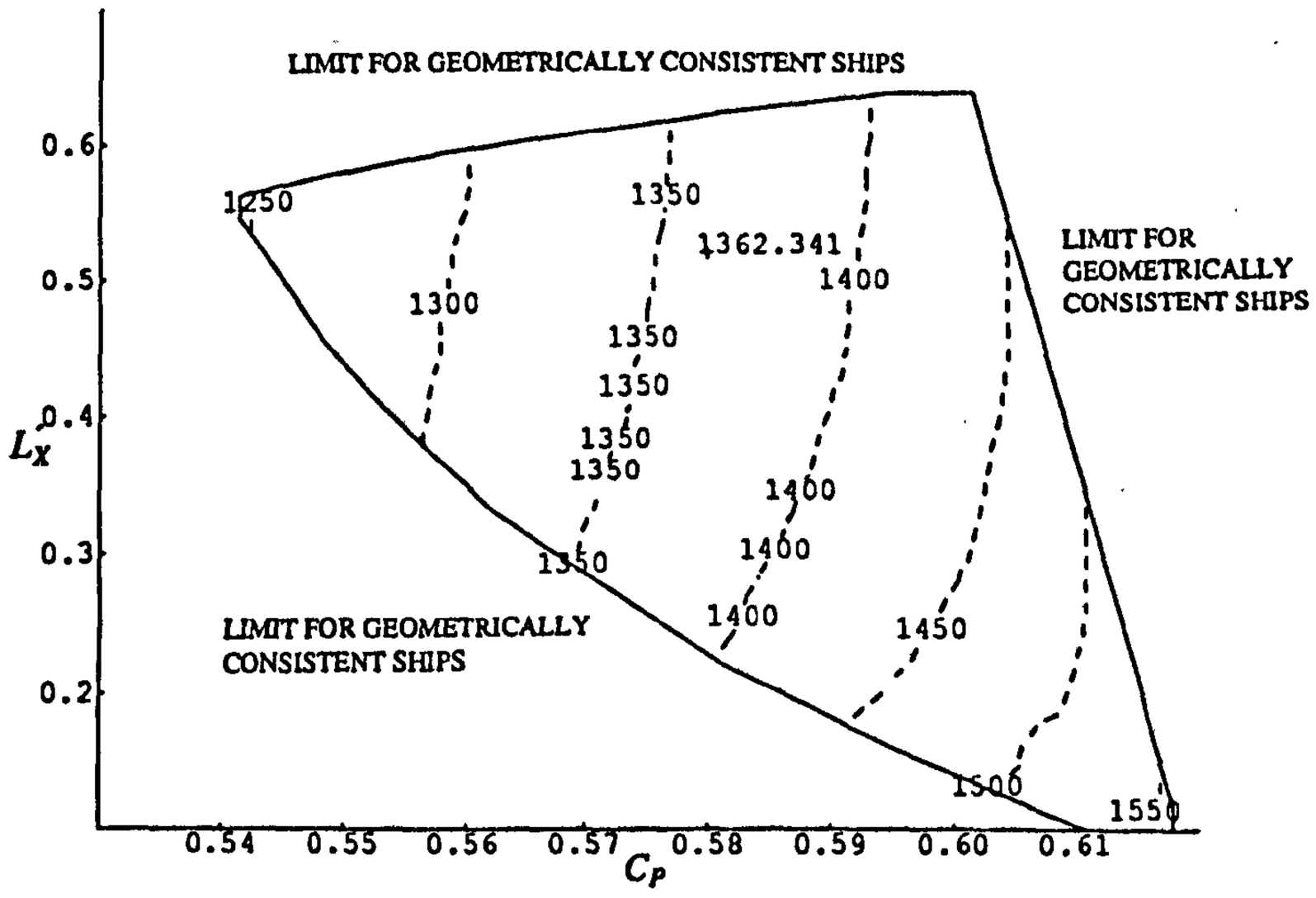

Figure 6.9a - Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L_X, at Ⓜ=7.0, unflared.

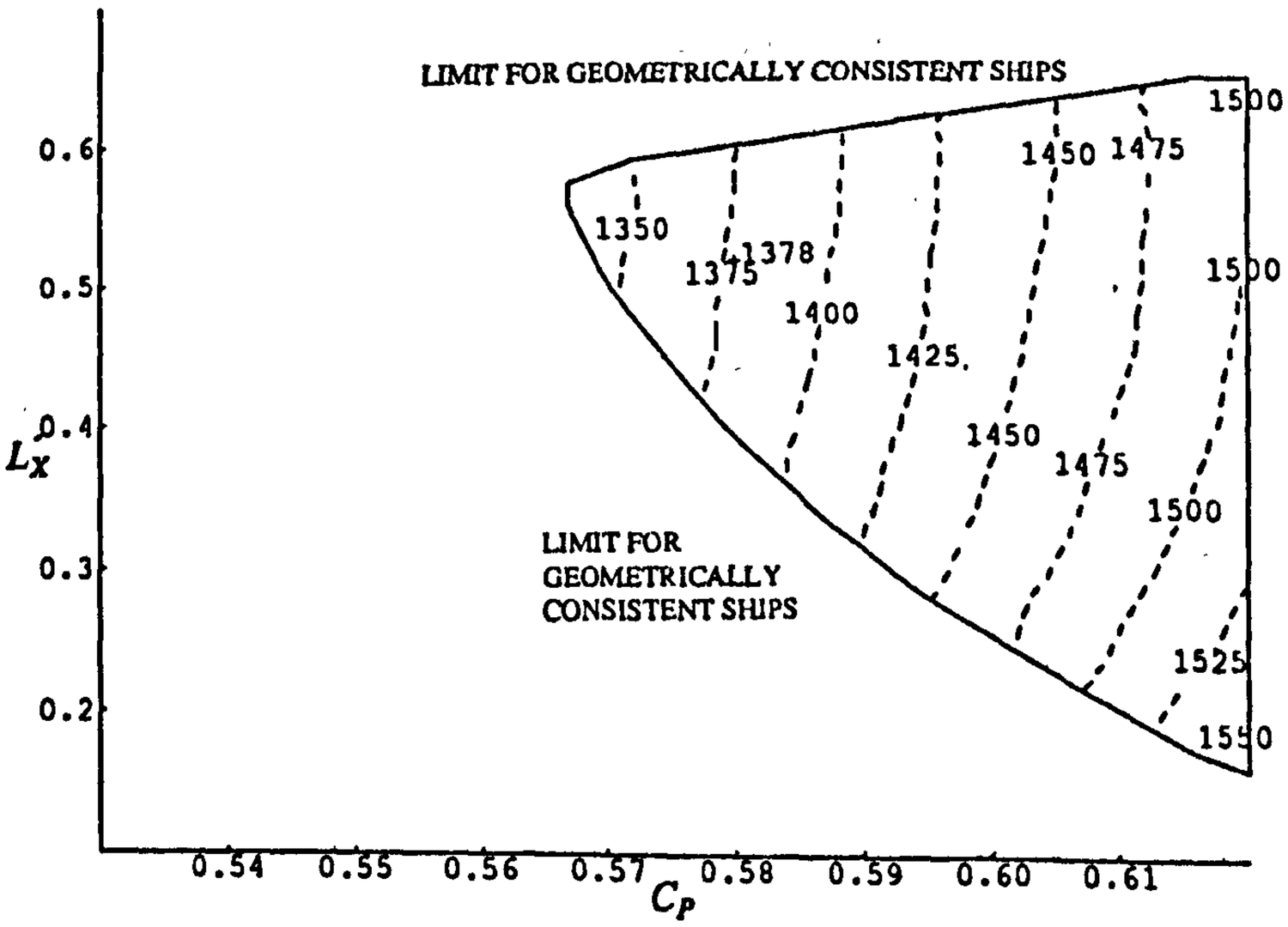

Figure 6.9b - Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L_X, at Ⓜ=7.0, flared.

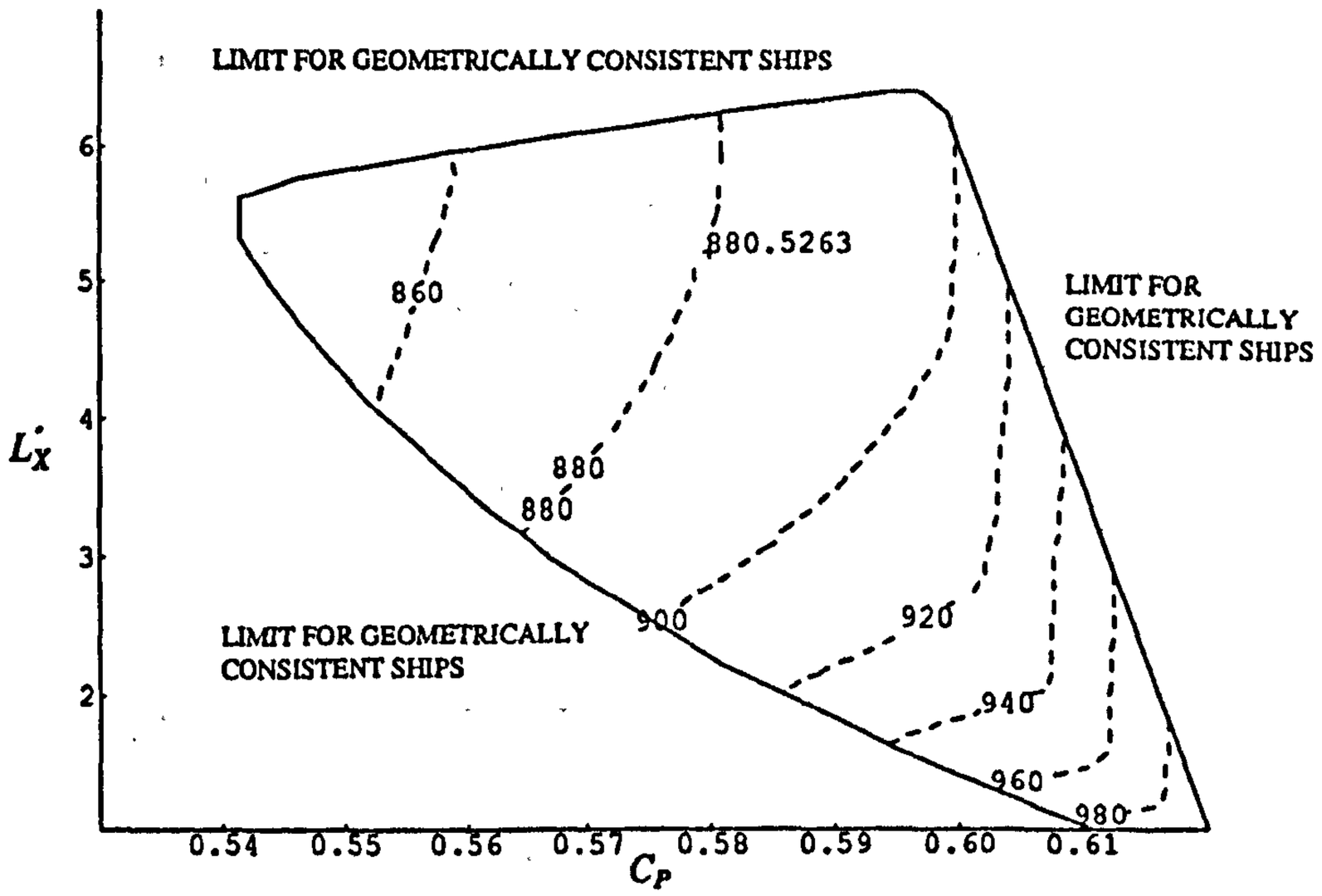

Figure 6.10a - Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L_X^*, at ⓜ=9.0, unflared.

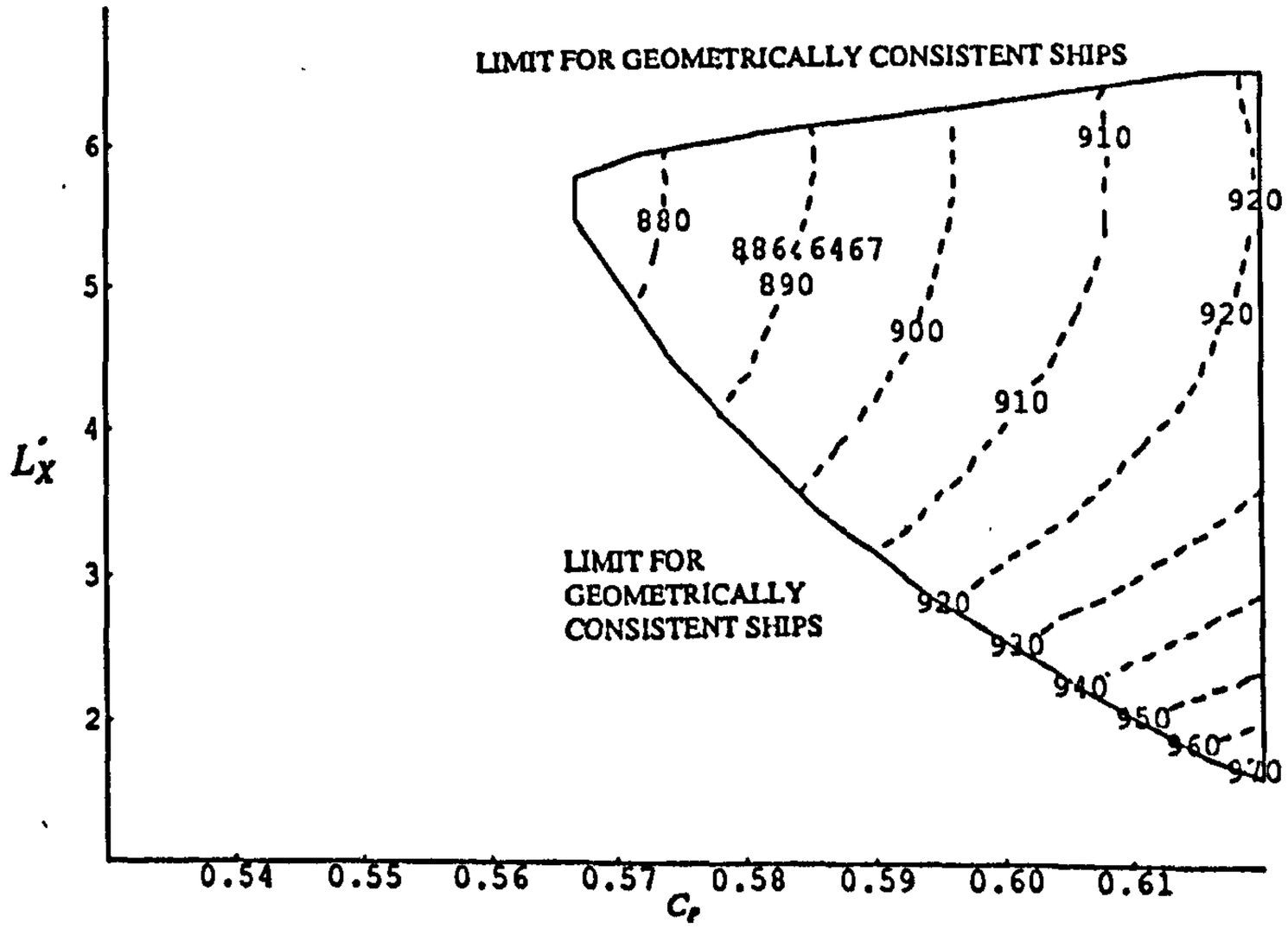

Figure 6.10b - Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L_X^*, at ⓜ=9.0, flared.

RUN NUMBER		1	2	3	4	5	6	7
Method		APPROX	APPROX	APPROX	APPROX	SEEK1	SEEK2	SEEK2
Rule Number		1	30	7	8	11	12	22
No. of Variables		2	5	5	5	5	5	5
Success/Failure		Success	Failure	Failure	Failure	Failure	Failure	Success
Number of Loops		22	41	6	13	611	1217	323
Objective Function	R_T (KN)	1050.80	858.35 †	-	854.84 $	857.67 *	856.41	856.79
Trial Vector	ⓜ	8.453	9.037		9.067	8.997	8.947	8.990
	B_{WL}/T	4.315	3.532		3.504	3.504	3.550	3.496
	C_P		0.5550	-	0.5545	0.5535	0.5486	0.5527
	$FLARE'_X$	-	5.250		5.355	5.250	2.629	5.330
	L'_X		0.566		0.571	0.576	0.575	0.576
Constraint Vector	Initial GM (m)	2.40	1.07		1.02	1.03	1.05	1.02
	Area 0-40° (m rad)	0.57	0.33		0.32	0.33	0.32	0.32
	Area 30-40° (m rad)	0.24	0.16		0.16	0.16	0.15	0.16
	Area 0-30° (m rad)	0.331	0.171	-	0.165	0.166	0.165	0.165
	Angle Max GZ (°)	46	53		53	53	52	53
	Max GZ (m)	1.55	1.30		1.28	1.29	1.19	1.29
	L_{WL}/D	14.0	13.99		14.02	13.85	13.83	13.82
	$\nabla_{ENC}(m^3)$	17000	17618		17615	17589	17010	17598

Table 6.4 - Optimization using Knowledge Base 2.

† results taken from loop 40, before failure.
$ results taken from loop 9, before failure.
* loop 15 - condition selected for runs nos. 6 and 7.

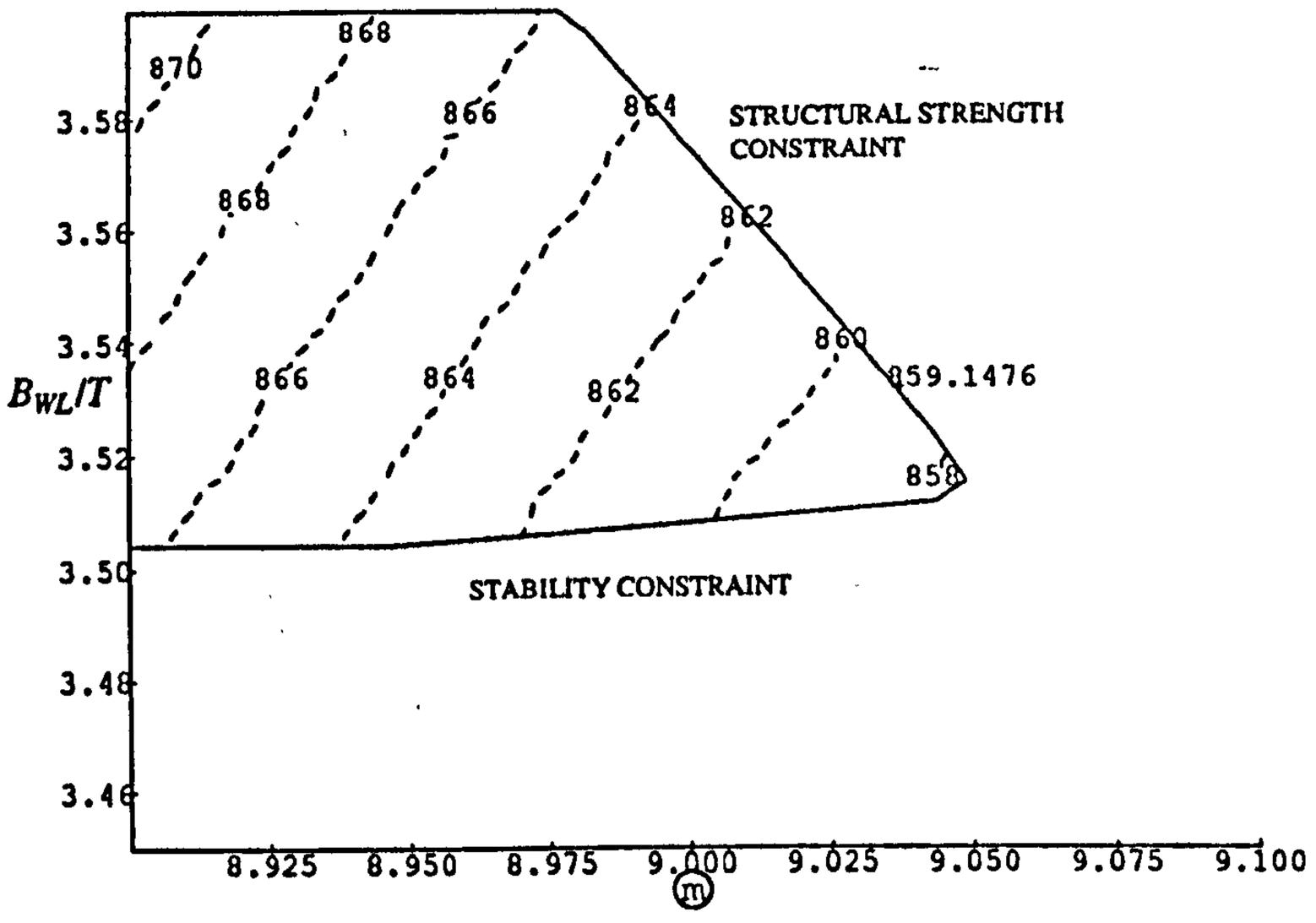

Figure 6.11 - Detail contour map of total resistance, R_T (kN) at 30 kts. for length to displaced volume ratio, (m) versus waterline beam to draught ratio, B_{WL}/T, at the "best found" region.

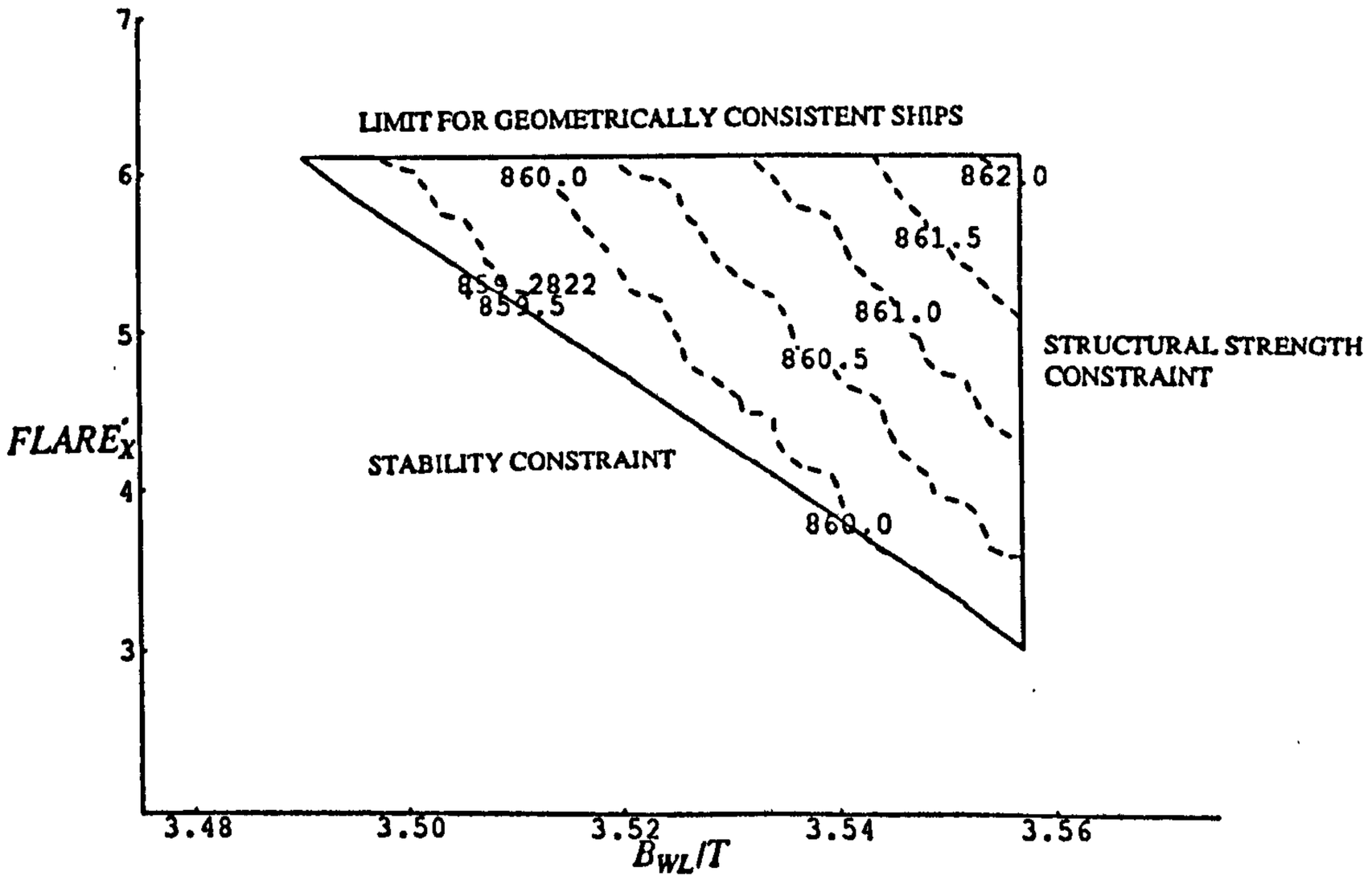

Figure 6.12 - Contour map of total resistance, R_T (kN) at 30 kts. for waterline flare at maximum beam, $FLARE_X^{\cdot}$ versus waterline beam to draught ratio, B_{WL}/T, based on a ship with rounded up characteristics at the "best found" region.

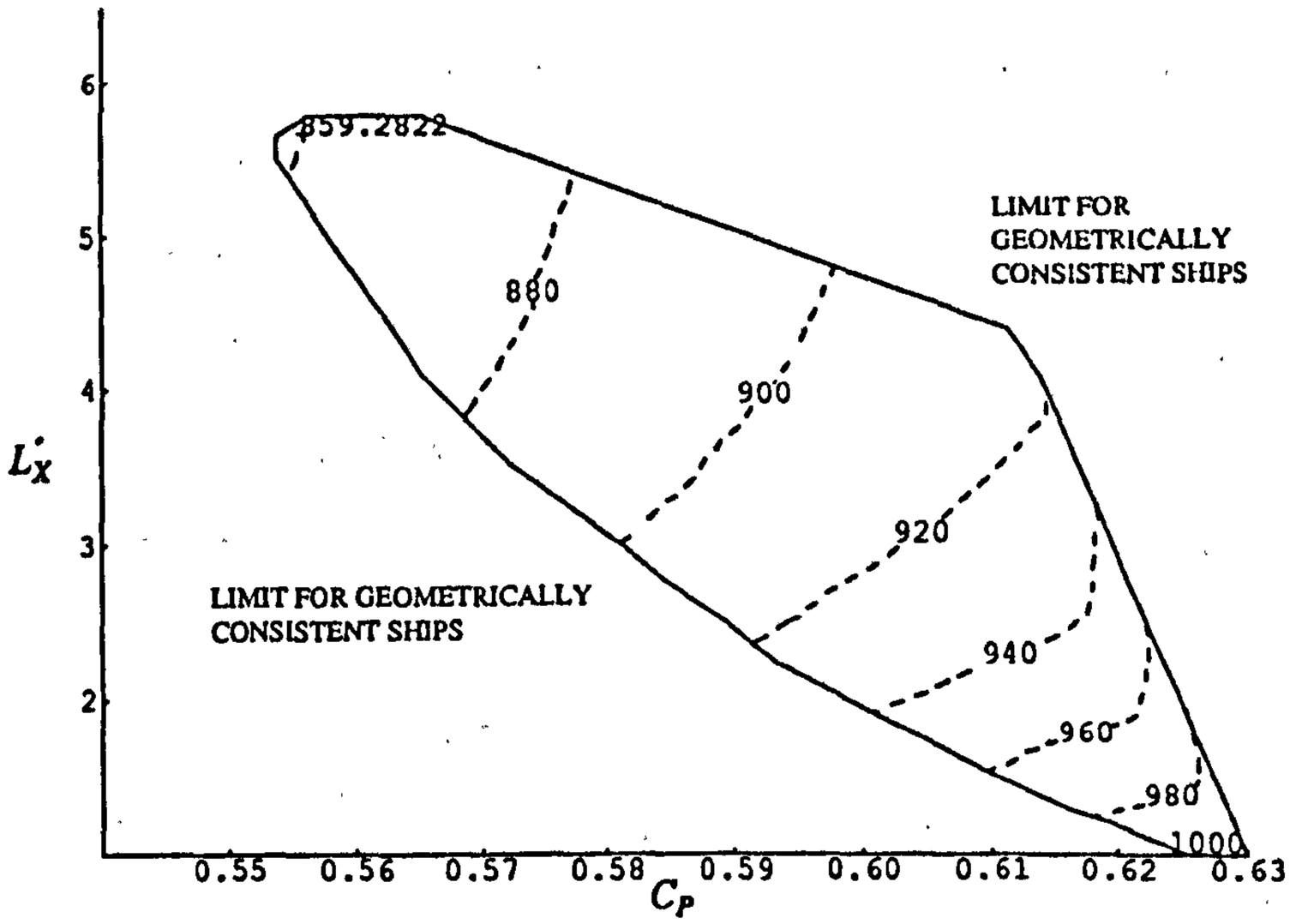

Figure 6.13 - Contour map of total resistance, R_T (kN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L_X^*, based on a ship with rounded up characteristics at the "best found" region.

7. CONCLUSIONS

An integrated computational approach to Ship Concept Design using optimization techniques and a knowledge base to control the optimization process has been presented. The system automates both synthesis and analysis; analysis by the repeated sequential use of Design Theory Modules and synthesis through the optimization process, which compromises conflicting requirements, subject to constraints.

The intention of this work has been to find a better approach to automated design synthesis and at the same time to employ detailed analytical tools such as a three-dimensional hull-form definition (CAD) and engineering analysis modules (CAE). To undertake this work a design system structure was developed to simulate the way a naval architect reasons and controls the traditional design process carried out in a design office. The design system produced adopts naval architectural theory to perform calculations and tries to achieve a balance of the inevitably conflicting results, presenting as the final configuration those data which produced the best compromise. The system is modular and flexible, thus allowing for the user's participation in the design process either by running it in a manual mode, or by prescribing trial vectors, boundaries and objective functions or even to modify, add or substitute theory and criteria, e.g., for different sorts of ships or requirements or for different design problems.

The data-base structure used in this system forms a central store of design information and separates the processes which are not design specific from data which relate only to the current task. The system allows the rapid testing of new ideas and the examination of particular features to establish their influences. It is structured to allow repeated cycling of the Design Theory Modules through a Control Module that non-dimensionalises geometrical parameters from a basic set to ensure consistency with each change made. This feature allows the theory that deals with geometry, such as hull-form definitions and general arrangements to become suitable for automation. It also allows for the use of default values of parameters, which allows for detailed theoretical calculations to be made at an early stage.

This automation concept opens the way to various synthesis processes: the one adopted in this work being optimization. Optimizers can perform full design loops of the system, under control of an optimization control device that varies parameters according to the strategy of the method deployed, starting from some initial design selection. This would normally be done in conjunction with manually carried out designs.

The advantages of using optimization are that the system can make use of the benefits that goal oriented tasks can provide, i.e., targets and restrictions can be specified and improvements driven by measures of merit. Such a structure allows for exploratory searches, which examine competitive ideas and search for successful combinations. This makes innovation more likely, because the searches test previously untried combinations of parameters and combine the beneficial tendencies.

To allow the system to take advantage of this approach, effort has been concentrated on overcoming problems that have made optimizers unattractive in the past. The principal problem of the uncertainty a designer feels when using optimization techniques is lessened by using a mapping strategy and this was found to be crucial to help with n-dimensional visualisation.

To aid in this process a knowledge base was developed to give expert advice on how to perform the mapping tasks. This also tries to reduce time consumption during optimization, another major stumbling block of optimization, by giving advice on how to look for promising regions of the design space. To achieve this the knowledge base developed gives advice on the selection of trial vectors, boundaries and objective functions and how to map them according to the mapping strategy developed. It also gives advice on the uses of the optimization methods adopted, following a hierarchy of utilisation and tackling the various modes of failure, starting with a reduced, two variable problem and proceeding to n dimensions.

Two examples have been used to illustrate this work based on the design of a frigate, one using optimization (Chapter 4) and the other controlling the optimization process with a knowledge base (Chapter 6). It was found that optimization can bring successful results but expertise is required to use it well. When this expertise is applied, the results are found more quickly and with increased confidence. It was also found that design searches should aim for regions of good solutions rather than exact points. The final mappings of resultant

regions show the multiple-parametric role of both the objective functions and the constraints. Using the knowledge base the user need not be expert in optimization theory or practice, instead having his time freed for ship design considerations. Good design theory modules were also found to be desirable to allow the system to simulate reasonably well the synthesis process. Nevertheless, good design insight, knowledge of design and the ability to recognise good results are still required to get the best from the system.

As a whole it can be seen that optimization has a role to play in ship design and can bring advantages but must be well controlled. Expert system structures can be very useful for design systems as long as they are not used to impose knowledge in the design subject (and this is ensured by keeping the theory out of the rule base), but are instead used for the design process and specific narrow domains of design techniques, leaving the synthesis process to be result-oriented. In other words, it is desirable to have a system where the knowledge base controls the synthesis process and the techniques that manipulate the system rather than any design specific decisions.

The principal contributions that this work has provided are:-

(1) The automation of the naval architectural synthesis process, allowing repeated cycles through design calculations.

(2) The ability for non-experts to have access to a sophisticated goal oriented search process using an expert system.

(3) Improved solution speeds for the optimization task using a multilevel, expert system controlled, approach to the problem.

As might be expected, several improvements to the system are still required and there are many avenues open for future work and research. The knowledge base is still a small one and needs to have more rules inserted, but the existing structure can be maintained. In the immediate future some extra warnings and mapping advice need to be added. Different optimization techniques ought to be tried and the implications of their use analysed. A much more complete set of naval architectural theory modules would provide the means of further testing and perhaps require adjustments for a version that would be workable for real designs in design offices.

APPENDIX A - OPTIMIZATION METHODS (OPTIVAR)

All the optimization methods used here try to solve a problem which may be described in mathematical terms as follows:-

find

$$U = U\ (x_1, x_2\ \ldots, x_n) = \text{minimum}$$

subject to

$$\Psi_i\ (x_1, x_2, \ldots, x_n) = 0 \qquad i = 1,m$$

and

$$\phi_j\ (x_1, x_2, \ldots, x_n) > 0 \qquad j = 1,p$$

where U is the objective function, $x_1, \ldots, x_n$ are the n variables that may be changed by the optimizer (the trial vector), Ψ_i (i=1,m) are m equality constraints and ϕ_j (j=1,p) are p inequality constraints. (n.b. random searches cannot deal with equality constraints, but these are not applicable here).

A.1. Method of Successive Linear Approximation (APPROX)

The method employed is a successive linear approximation developed by Griffith and Stewart. Starting at an initial feasible point ($x_1^o, x_2^o, \ldots, x_n^o$) the functions are approximated by expansion in a Taylor's series about x_i^o in which higher order terms are dropped. As the problem has been converted to linear form, it can be solved by linear programming. The linearisation is applied over a finite range that is subsequently moved and reduced as the optimization proceeds. The constraints, which

are also linearised, are used to further reduce the linearised region. This method is very fast but has final convergence problems. These problems are exacerbated by singularities or excessive constraints.

A.2. Random Exploration with Shrinkage (RANDOM)

This method consists of a random search for the minimum, using a shotgun technique, with iterative shrinkage. Random points for each variable x_1 to x_n are generated from the expression $x_i = l_i + r_i (u_i - l_i)$ where l_i is the estimated lower limit for x_i,u_i is the estimated upper limit to x_i and r_i is a random number uniformly distributed between zero and one. Any generated point that violates an inequality constraint is discarded before the objective function is evaluated. If the constraints are violated a certain number of times consecutively, by default 300, the process stops.

The search is begun by evaluating a number of random points using the above equation, this number being a multiple of the number of variables. From these, the best results are selected and used as the basis for a new and shrunken range for each variable. The number of best results is defined by dividing the number of random points by a shrinkage factor. Within this new space, new random points are evaluated. These and the previous best results are sorted to yield a new set of best results and a new shrunken space. The process is repeated until the range of each variable is acceptably small, or until the range has been shrunken a certain number of times, 1000 being the default. If any set of random points does not include one from near the optimal region the shrinkage mechanism can reject the area around the optimum and thus carries the

risk of finding a false optimum. However, this mechanism does significantly improve the speed of the search, particularly over high dimensional spaces. When the problem is highly constrained shrinkage must be treated with caution.

A.3 Hooks and Jeeves Direct Search (SEEK)

This method uses the following heuristic algorithm for minimisation:-

(1) An arbitrary starting point must be selected, called the initial base point, at which the objective function U is evaluated.

(2) An exploratory search is begun by increasing x_1 by a predetermined step length. If this improves the value of U, it is retained, if not, a corresponding negative step in taken. If both fail, no step is taken. A similar exploration is made for x_2 and so on. The final result of such an exploratory search is called a base point.

(3) A pattern move is made by changing each variable from the last base point by an amount equal to the difference between the new and previous base point. This difference will commonly include a previous pattern move.

(4) If the pattern move fails to improve U, it is cancelled and replaced by a new search. If it succeeds, it is followed by a new search.

(5) The iteration continues until the exploratory search fails to locate a better point. The step length is then reduced by an arbitrary amount and the search is repeated. After each failure the step length is reduced an additional fraction until it reaches some predetermined minimum, when it is assumed that the optimum has been reached.

A.4. Penalty Functions

A penalty function is a simple technique used to distort the optimization function in order to force the search towards feasibility, when a constraint is, or is about to be, violated. The simplest form of penalty function is:

$$U_p = U(x_1, x_2, \ldots, x_n) + r \sum_{i=1}^{m} |\Psi_i| + r \sum_{j=1}^{p} |\langle\phi_j\rangle|$$

where U_p is the penalised objective function which is minimised instead of U, r is a large number and the symbol $\langle\alpha\rangle$ has the meaning

$$\langle\alpha\rangle = \begin{cases} \alpha \text{ if } \alpha \leq 0 \\ 0 \text{ if } \alpha > 0 \end{cases}$$

This penalty function is used as the one pass external function with $r = 10^{20}$. This is very severe and sometimes the search stalls, particularly if it must follow along a constraint line to reach the optimum. Despite this disadvantage, the strategy is often quite successful, and requires

a minimum of computer time when it works. (When using this function, the SEEK algorithm uses 100 random shotgun steps when it fails to find a better point, to overcome stalling).

The other penalty functions in the OPTIVAR suite soften this severe distortion and to try to achieve less risk of premature stalling of the search. The one pass penalty function is only active when constraints are violated and the search is in an infeasible region. Functions having this characteristic are called external penalty functions. Conversely, internal functions are active only in the feasible region, so that the surface is distorted when the inequality constraint line is approached, and the search is 'warned' that there is trouble ahead. The Fiacco-McCormick function combines these approaches and takes the form:-

$$U_p = U(x_1, x_2, \ldots, x_n) + (1/r)\sum_{i=1}^{m}\Psi_i^2 + r^2\sum_{j=1}^{p}1/\phi_j^{\alpha} + (1/r)\sum_{j=1}^{p}\langle\phi_j\rangle^2$$

Here the symbol <> has the previous meaning indicating an unsatisfied constraint, while superscript S indicates a satisfied one. The process begins with r=1 and a sequence of optimization problems are solved in which r is progressively reduced. To justify this expression, consider the effect on a two-dimensional problem with one inequality constraint. If the interior term is considered, (r^2/ϕ_1^{α}), with r=1, it will push the surface upwards asymptotically to the constraint line, beginning the distortion some distance into the feasible region. As the constraint line is approached ϕ_1^{α} becomes smaller and smaller. It is apparent that a false constrained optimum point will be established well away

from the true location at the constraint line. As r becomes smaller, the effect of $1/\phi_1^m$ is increasingly reduced, the surface is sharpened, and the false constrained optimum point will approach the true one.

The external term $\langle\phi_1\rangle^2/r$ has a similar effect in the exterior region. With r=1, there is a gentle distortion of the surface remote from the constraint line which becomes small as ϕ_1 approaches zero at the constraint line. A false unconstrained optimum point is created near the feasible region. Subsequent reductions in r increase the effect of the penalty term, sharpening the curve, and shifting the false, unconstrained optimum toward the true constrained location.

APPENDIX B

Knowledge Base 2 (rule base only)

```
 1 : /* SETTING UP and METHOD SELECTION
 2 :
 3 : /* R 1
 4 : if method is APPROX
 5 : then run linear_approx (operation);
 6 : cycle_limit = 100;
 7 : optimization is done
 8 :
 9 : /* R 2
10 : if method is SEEK_1
11 : then run direct_search1 (operation);
12 : optimization is done
13 :
14 : /* R 3
15 : if method is SEEK_2
16 : then run direct_search2 (operation);
17 : optimization is done
18 :
19 : /* R 4
20 : if method is RANDOM
21 : and n_dim is done
22 : then cycle_mode is stop;
23 : run random_search ();
24 : goal is done
25 :
26 : /* THE APPROX FAILURE BRANCH
27 :
28 : /* R 5
29 : if optimization is done
30 : and method is APPROX
31 : and optimize is failure
32 : and reason is 'infeasible point'
33 : then cycle_mode is autocycle;
34 : run tune(operation) ;
35 : goal is done
36 :
37 : /* R 6
38 : if optimization is done
39 : and method is APPROX
40 : and optimize is failure
41 : and closer_startpoint is not tried
42 : and reason is 'no convergence'
43 : and linearity is no
44 : then run proc_seek (operation);
45 : cycle_mode is autocycle;
46 : method is SEEK_1;
47 : goal is done
48 :
49 : /* R 7
50 : if optimization is done
51 : and method is APPROX
52 : and optimize is failure
53 : and closer_startpoint is not tried
54 : and reason is 'no convergence'
55 : and linearity is yes
56 : then cycle_mode is autocycle;
57 : run check_plane(operation);
```

```
58 : closer_startpoint is tried;
59 : goal is done
60 :
61 : /* R 8
62 : if closer_startpoint is tried
63 : and tune_steps is not tried
64 : and optimization is done
65 : and method is APPROX
66 : and optimize is failure
67 : and reason is 'no convergence'
68 : then cycle_mode is autocycle;
69 : run tune_step_features(operation);
70 : tune_steps is tried;
71 : goal is done
72 :
73 : /* R 9
74 : if tune_steps is tried
75 : and closer_startpoint is tried
76 : and optimization is done
77 : and method is APPROX
78 : and optimize is failure
79 : and reason is 'no convergence'
80 : then run isolate(operation,improvement);
81 : isolation is done
82 :
83 : /* R 10
84 : if isolation is done
85 : and improvement is yes
86 : then run approx_from_random(operation);
87 : cycle_mode is autocycle;
88 : goal is done
89 :
90 : /* R 11
91 : if isolation is done
92 : and improvement is no
93 : then run proc_seek(operation) ;
94 : cycle_mode is autocycle;
95 : method is SEEK_1;
96 : goal is done
97 :
98 : /* THE SEEK_1 FAILURE BRANCH
99 :
100 : /* R 12
101 : if optimization is done
102 : and method is SEEK_1
103 : and optimize is failure
104 : and cause is 'hang up'
105 : and used_all_shots is yes
106 : then cycle_mode is autocycle;
107 : run change_PF (operation,hang_up1,infeas1);
108 : method is SEEK_2;
109 : goal is done
110 :
111 : /* R 13
112 : if optimization is done
113 : and method is SEEK_1
114 : and optimize is failure
115 : and cause is 'hang up'
116 : and used_all_shots is no
```

```
117 : and hang_up1 is yes
118 : then cycle_mode is autocycle;
119 : run diff_start (operation);
120 : hang_up1 is no;
121 : goal is done
122 :
123 : /* R 14
124 : if optimization is done
125 : and method is SEEK_1
126 : and optimize is failure
127 : and cause is 'hang up'
128 : and hang_up1 is no
129 : then cycle_mode is autocycle;
130 : run change_PF (operation,hang_up1,infeas1);
131 : method is SEEK_2;
132 : goal is done
133 :
134 : /* R 15
135 : if optimization is done
136 : and method is SEEK_1
137 : and optimize is failure
138 : and cause is 'no convergence'
139 : then cycle_mode is autocycle;
140 : run change_PF (operation,hang_up1,infeas1);
141 : method is SEEK_2;
142 : goal is done
143 :
144 : /* R 16
145 : if optimization is done
146 : and method is SEEK_1
147 : and optimize is failure
148 : and cause is 'no feasible sol.'
149 : and infeas1 is yes
150 : then cycle_mode is autocycle;
151 : run relax_cons (operation);
152 : infeas1 is no;
153 : goal is done
154 :
155 : /* R 17
156 : if optimization is done
157 : and method is SEEK_1
158 : and optimize is failure
159 : and cause is 'no feasible sol.'
160 : and infeas1 is no
161 : then cycle_mode is autocycle;
162 : run change_PF (operation,hang_up1,infeas1);
163 : method is SEEK_2;
164 : goal is done
165 :
166 : /* THE SEEK_2 FAILURE BRANCH
167 :
168 : /* R 18
169 : if optimization is done
170 : and method is SEEK_2
171 : and optimize is failure
172 : and cause is 'hang up'
173 : and used_all_shots is yes
174 : then cycle_mode is autocycle;
175 : run change_method (operation);
```

```
176 : method is RANDOM;
177 : goal is done
178 :
179 : /* R 19
180 : if optimization is done
181 : and method is SEEK_2
182 : and optimize is failure
183 : and cause is 'hang up'
184 : and used_all_shots is no
185 : and hang_up1 is yes
186 : then cycle_mode is autocycle;
187 : run diff_start (operation);
188 : hang_up1 is no;
189 : goal is done
190 :
191 : /* R 20
192 : if optimization is done
193 : and method is SEEK_2
194 : and optimize is failure
195 : and cause is 'hang up'
196 : and hang_up1 is no
197 : then cycle_mode is autocycle;
198 : run change_method (operation);
199 : method is RANDOM;
200 : goal is done
201 :
202 : /* R 21
203 : if optimization is done
204 : and method is SEEK_2
205 : and optimize is failure
206 : and cause is 'no convergence'
207 : then cycle_mode is autocycle;
208 : run change_method (operation);
209 : method is RANDOM;
210 : goal is done
211 :
212 : /* R 22
213 : if optimization is done
214 : and method is SEEK_2
215 : and optimize is failure
216 : and cause is 'no feasible sol.'
217 : and infeas1 is yes
218 : then cycle_mode is autocycle;
219 : run relax_cons (operation);
220 : infeas1 is no;
221 : goal is done
222 :
223 : /* R 23
224 : if optimization is done
225 : and method is SEEK_2
226 : and optimize is failure
227 : and cause is 'no feasible sol.'
228 : and infeas1 is no
229 : then cycle_mode is autocycle;
230 : run change_method (operation);
231 : method is RANDOM;
232 : goal is done
233 :
234 : /* R 24
```

```
235 : if method is RANDOM
236 : and n_dim is not done
237 : then run comment_function (operation);
238 : run check_optimum(opt_analysis);
239 : check is done
240 :
241 : /* THE SUCCESS BRANCH
242 :
243 : /* R 25
244 : if optimization is done
245 : and optimize is success
246 : and n_dim is not done
247 : then run check_optimum(opt_analysis);
248 : check is done
249 :
250 : /* R 26
251 : if check is done
252 : and opt_analysis is 'no visible minimum'
253 : then run amplify_search_area(search);
254 : new_mapping is done
255 :
256 : /* R 27
257 : if new_mapping is done
258 : and search is 'only one region'
259 : then amplification is done
260 :
261 : /* R 28
262 : if new_mapping is done
263 : and search is 'more than one region'
264 : then amplified_search is done
265 :
266 : /* R 29
267 : if check is done
268 : and opt_analysis is 'one of minimums'
269 : or amplified_search is done
270 : then run manual_ship_tests(operation);
271 : false_optima is sorted_out
272 :
273 : /* R 30
274 : if check is done
275 : and opt_analysis is 'only visible minimum'
276 : or amplification is done
277 : or false_optima is sorted_out
278 : then cycle_mode is autocycle;
279 : run set_n_dim(operation,closer_startpoint,tune_steps,hang_up1,
280 : n_dim is done;                                    infeas1);
281 : goal is done
282 :
283 : /* THE PUNCHLINE
284 :
285 : /* R 31
286 : if optimization is done
287 : and optimize is success
288 : and n_dim is done
289 : then cycle_mode is stop;
290 : goal is done
291 :
292 : seek goal
293 :
```

Knowledge Base 2 - LIST OF OBJECT FRAMES/PROCEDURES

1	method	Text
2	linear_approx	Procedure
3	operation	Text
4	cycle_limit	Real
5	optimization	Text
6	direct_search1	Procedure
7	direct_search2	Procedure
8	n_dim	Text
9	cycle_mode	Text
10	random_search	Procedure
11	goal	Text
12	optimize	Text
13	reason	Text
14	tune	Procedure
15	closer_startpoint	Text
16	linearity	Text
17	proc_seek	Procedure
18	check_plane	Procedure
19	tune_steps	Text
20	tune_step_features	Procedure
21	isolate	Procedure
22	improvement	Text
23	isolation	Text
24	approx_from_random	Procedure
25	cause	Text
26	used_all_shots	Text
27	change_PF	Procedure
28	hang_up1	Text
29	infeas1	Text
30	diff_start	Procedure
31	relax_cons	Procedure
32	change_method	Procedure
33	comment_function	Procedure
34	check_optimum	Procedure
35	opt_analysis	Text
36	check	Text
37	amplify_search_area	Procedure
38	search	Text
39	new_mapping	Text
40	amplification	Text
41	amplified_search	Text
42	manual_ship_tests	Procedure
43	false_optima	Text
44	set_n_dim	Procedure
45	cycle_counter	Real

APPENDIX C

PAPER No 11

THE ROYAL INSTITUTION OF NAVAL ARCHITECTS

Spring Meetings 1990

The issue of this copy of the paper is on the express understanding that an abstract only may be published after the paper has been read at the meeting to be held in the Weir Lecture Hall, 10 Upper Belgrave Street, London SW1X 8BQ, on April 25, 1990.

The Institution is not, as a body, responsible for the statements made or the opinions expressed by individual authors or speakers.

Written contributions to the discussion should reach the Secretary, The Royal Institution of Naval Architects before June 18, 1990.

OPTIMIZATION TECHNIQUES IN SHIP CONCEPT DESIGN

by A.J. Keanet (Member), W.G. Price* (Fellow) and R.D. Schachter*

SUMMARY This paper describes an integrated computational approach to ship concept design using optimization techniques. Although normally heavily automated, the approach used also allows for the designer's naval architectural knowledge and creativity to direct the design process. The method described incorporates accepted naval architectural tools, a sophisticated data-base handler and several optimization procedures. It is based on a modular construction and this provides the designer with the ability to modify or include a variety of analytical methods and data to suit the problem in hand. It also allows the designer to select appropriate goals for optimization and to prescribe limiting constraints. Using such a system the design process follows its normal course commencing from a small set of key parameters and proceeds towards a complete hull-form definition consisting of weight, space, offset data, etc. The advantages of employing optimization techniques in this process are discussed and contrasted with more traditional C.A.D. methods where all design decisions are retained by the user. The possible uses of expert systems in this role are also briefly addressed.

The philosophy and structure of the proposed approach are illustrated by its application to the preliminary design of a frigate hull. The example studied uses dual (e.g. beam/draught versus length to displaced volume ratio, ∇, say) and multi-parameter (e.g. beam/draught, ∇, flare, prismatic coefficient, etc.,) optimizations. Additionally, several different optimization procedures are outlined and their merits, failures, suitability, etc., are discussed in the context of ship concept design.

1. INTRODUCTION

The availability of powerful computing facilities has transformed the work of the average design engineer. New designs are now produced to greater levels of detail, in shorter time-scales and with fewer people than hitherto. Naval architecture has not escaped this revolution and most ship design organisations have access to software dealing with a wide variety of tasks. Typically, programs are available for dealing with hull fairing, hydrostatics, stability, damage, motions, strength, powering, manoeuvring, costing, etc. However, most of these programs are analytical rather than creative in nature, having been developed in isolation to deal with the specific requirements of an essentially numerate discipline. The application of computers to the synthesis of new designs has proven intractable. This is perhaps to be expected, given the difficulty of capturing the creative talent of the designer. This division between design synthesis and analysis has been identified by many authors; an extensive discussion in the context of preliminary ship design is given by Andrews[1,2]. Perhaps the most important problem identified in those works was the difficulty of deciding overall ship sizing parameters (such as length, beam, etc.,) at a stage of design when most of the analytical tools of naval architecture may not readily be applied. By way of example, consider the choice of beam when a hull-form is not available to calculate stability curves, or the choice of enclosed volume when the distribution of internal spaces has not yet been made. Traditionally, these problems have been overcome by applying various empirically based approaches, i.e. by assuming that initial meta-centric height will provide a good measure of stability or by using a compartment density figure to decide space requirements given estimates of overall displacement. These methods allow initial estimates of the various sizing parameters to be made and an iterative design spiral is then commenced, with increasing levels of detail allowing ever more refined techniques to be applied, until a final design is produced. To illustrate the advantages of computer based methods in this context Andrews[2] cites the use of a program called WSVPROG, developed by one of the authors[3], to point out that computerized design packages can find good designs that are not immediately obvious. The example mentioned there occurs when increases in size require increases in installed power which further drive up the required displacement. Andrews[2] notes that such trends can be reversed and that this 'feature is not normally revealed by non-interactive, non-computerised initial ship sizing'. However, as he goes on to note, any approach based on empirical formulae limits the ability of the designer to make fundamental design changes in the light of detailed design analysis carried out far down the design path; the investment in design development will be too costly to be able to afford such changes. This failure to reflect the results of the most accurate analysis in a design limits the usefulness of such methods and tends to act against innovative solutions. The only way around this problem seems to be to apply the established analytical tools at the earliest stages of design and it is on this area that design aids, as opposed to analytical tools, must be focused. They must allow the designer to have full access to the profession's complete tool-kit of analytical software, rather than

† Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, U.K.
formerly of
Department of Mechanical Engineering, Brunel University, Uxbridge, Middlsx., UB8 3PH, U.K.
* Department of Mechanical Engineering, Brunel University, Uxbridge, Middsx., UB8 3PH, U.K.

leaving these to be applied in an essentially remedial role late on in the design process. This early application of detailed analytical tools is the approach followed here, e.g., complete GZ curves for all conditions are produced and analysed.

A number of papers have appeared in the literature describing computer aided design systems that address this problem, see for example, Eames and Drummond[4], Yuille[5], Calkins[6] or Parsons and Beir[7]. These systems integrate a number of the available analytical tools around design data-bases; the analyses applied being concerned primarily with the use of mathematical models to produce further information (both geometric and numerical) to help extend the contents of the data-base. In such systems the expert performs the design, making the key decisions, leaving the computer to carry out supporting analytical calculations or to display and present the results. More recent developments have addressed systems that help, or even replace, the expert by making decisions and/or by searching for successful combinations of ideas. A division can be made in such systems by distinguishing between the application of professional experience and the introduction of innovative ideas, arguing that the decision making process requires reasoning and the use of knowledge, while the search for successful combinations of ideas draws on creativity. These two branches of the design process are respectively addressed by *expert systems* and *optimization* techniques, the former encompassing methods for storing knowledge while the latter tackle the problems of comparative testing of new ideas.

Recently, most interest seems to have been concentrated on expert systems and a large amount of work is being carried out in this area, see, e.g., Tong[8]. In the jargon of the literature, an expert system is a 'symbolic computation technology for solving real problems in a new way'. This is achieved mainly through the use of a rule-based, heuristic structure, i.e., a structure based on *IF (a rule) THEN (an action or an advice)*. Using this device, reasoning can be simulated or justified and expert knowledge can be captured, stored and used by less skilled operators, see for example, Alty and Coombs[9] or, in the field of naval architecture, MacCallum and Duffy[10]. A superficial study of the literature shows that there seems to be much less recent interest in applying optimization to naval architectural concept design, although it is by no means novel, see for example, Mandel and Leopold[11], Gallin[12], Parsons[13], Beier et al[14] or, more recently, Pantazopoulos[15].

Optimization techniques do not, at first, look as attractive as expert systems when the application is meant to simulate human intelligence, relying as they do, on essentially numerate analysis. However, it should be noted that innovation is more likely using such techniques (i.e., they test previously untried combinations of parameters), especially when compared to expert systems approaches which, by definition, tend to produce designs that are very similar to those of an experienced engineer (i.e., they are evolutionary in nature). Another particular strength of optimization methods lies in their ability to perform goal oriented tasks; targets can be specified and optimizers deployed so as to drive a design towards the desired goals. This is much more difficult to institute with a rule-base regime, primarily because such goals are usually specified numerically and in order to make such improvements, the requirement of calculating variations in some measure of merit arises. Usually the aim is to make certain parameters as large or as small as possible although conflicts often arise. Of course, once such a mechanism for improvement has been found it can be pursued until exhausted. This is precisely the function of optimization techniques, whereas knowledge based methods try to direct the design process by reference to previously successful design rules, which are often non-numerate, natural language relationships. The exploratory search of an optimizer is of course a much longer process, because it tries possibilities regardless of whether human reasoning would recognize them as absurd. Usually, knowledge based structures lead more directly to the best design, also they are capable of indicating why various design decisions have be taken. The only justification open to optimizers on the other hand, is that the final design meets the specified requirements better than all the other possibilities tested. However, if the design process is to encompass the most reliable analytical tools the resulting design problem becomes one of great complexity, involving many subtle interdependencies. Under such circumstances a knowledge based structure begins to need simplifications and to require the insertion of new rules, i.e., more knowledge. Moreover such rules are difficult to make universal as problems grow in complexity, requiring new rules for each new design problem. In summary, no one approach is likely to provide all the answers, each having their own peculiarities, i.e. :-

(1) traditional C.A.D. systems are very flexible leaving the designer complete freedom in the decision making process but requiring great skill to use well;
(2) optimization methods are capable of extracting the most accurate information from the available design theory and applying it to meet specific goals but can be very time consuming to apply and difficult to understand;
(3) expert systems neatly distill the available knowledge and apply it in a fully explained fashion, even though the knowledge may need to be simplified to make it conform to the structure in use.

The best approach ought to encompass a combination of these ideas, i.e., to use exploratory searches, with guidance provided by an expert system using simple rules; the whole system being managed by the user via a powerful and flexible interface. This would allow the designer to vary both the measure of merit and/or the parameters used to achieve a good design as the design advances, or even to take manual control.

The aim of the present programme of work is to study the use of these 'modern' design techniques in naval architecture with the goal of providing a further step in computerizing the design process. With this in mind, a ship concept design system called OPTIONS has been developed. It currently uses optimization techniques to control suitable variables in an attempt to improve a design, subject to specific constraints. A rule-based decision making structure has been found necessary to improve control over certain parts of this process. The structure of OPTIONS has been designed to allow the integration of many stages of the naval architectural design spiral. It uses a modular construction that allows great flexibility in the choice of theory, through selectivity in the sub-programs employed, and in selecting data for the various parameters, variables, constraints, limits, design objectives, etc., through an internal data-base. By varying the selection and ordering of the various design theory modules, different ship types and design strategies can be investigated.

The present paper describes some of this work, discussing the fundamental concepts and overall principles of the application of computerized optimization techniques to preliminary ship design. *No* attempt is made to describe the details of the individual software algorithms used in the process. However, to assist the reader some background information is included and this is

given in the following section. The various modules involved are drawn from a number of sources although they are *not* exhaustive by any means. In fact, the design theory modules chosen here are just sufficient to demonstrate the ship concept design process; to be of practical use they would need to be added to, modified or even replaced to suit the requirements of the user and the task in hand. It is the overall arrangement and structure of the approach which is important and is the object of study. A subsequent publication will describe similar work currently being undertaken in the application of expert systems.

The OPTIONS system has been developed using the UNIX operating system on Sun 3/50 workstations. It is written in *ANSI 77 Fortran, and requires around 1.5 megabytes of memory* for processing. The system is fully portable, and can run on any UNIX based computer, supporting ANSI 77 Fortran and having the required memory. Graphical output makes use of the widely known GINO[16] and SIMPLEPLOT[17] packages.

2. DESIGN SYSTEM STRUCTURE

The structure of the OPTIONS design system is illustrated in Figure 1. The basic idea has been to develop a flexible framework together with a small number of components to allow satisfactory designs to be performed whilst evaluating the techniques employed. The existence of certain modules, even in a crude form, is necessary to ensure a balance between the competing aspects of naval architecture, enabling the study of realistic problems. Whenever possible, well recognized and accepted theories have been adopted. A brief description of these elements is given in the following sub-sections.

The current system consists of a *Design Control Module*, an *Optimizer*, the *Design Theory Modules*, a *Data-base Handler* and some *Auxiliary Routines*. The design theory modules currently available cater for hull-form creation and drawing, hydrostatic and stability curve calculations, stability assessment, enclosed volume estimation, light weight estimation and resistance calculation. It should be noted that some fundamental design aspects are still not represented, such as cost, seakeeping, structures, etc. The use of costing information would be particularly worthwhile for making more realistic evaluations of the optimization process. However, lack of published data has prevented the development of such a module to date. When one becomes available its natural role would be to form part of the objective function used as a goal by the optimizers.

2.1. Design Control Module (CONSST)

The Design Control Module forms the heart of the system. It is this routine which controls the order and choice of the design theory modules and auxiliary routines. Within this routine, ship parameters from the data-base are made consistent and form parameters non-dimensionalized, using a rule-based structure. These features allow parameters to be reduced to a numerically consistent form when initial data are given in an inconsistent way or after one or more variables have been modified. During this process extraneous data are overwritten, following a hierarchical rule-base, currently using Ⓜ, i.e., length to displaced volume ratio, beam to draught ratio and C_P, with C_B and displaced volume fixed for each run. Of course, like any other rule-based structure, this hierarchy could easily be changed and designs could then be made for fixed length and beam, say. Usually in a given design problem, some of the key parameters will be fixed, or at least based on previous practice; all that is required here is that the various parameters be made mutually consistent and it is this function that the rule-base addresses.

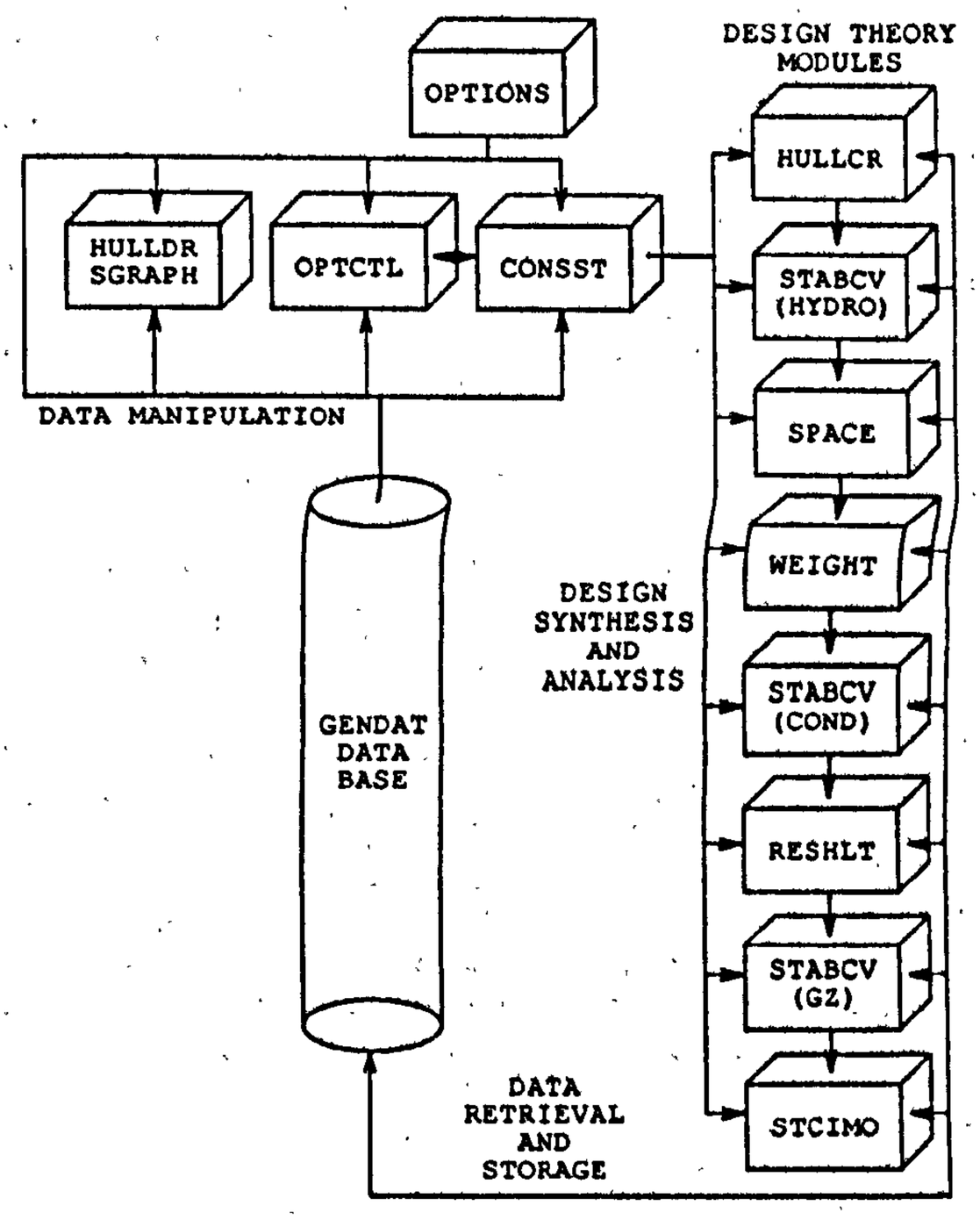

Figure 1 - Design system structure.

The Design Control Module can be used in one of two modes, both accessing the same naval architecture theory and a common data-base, i.e. :-

(1) Fully manual mode - all design decisions are taken by the user and individual theory modules are selected manually to support this process. This mode is similar to many existing naval architecture design tools.

(2) Automated mode with optimization - a few key decisions are taken by the designer to select the required goal and theories to be used. Optimization techniques are then employed to drive the design towards the desired goal with the theory modules being deployed by the optimizer only as and when required.

Since both methods use common theory and data, either can be used to reach the same design and this provides an important check on any solutions suggested by the optimizers. Having access to a number of powerful optimization routines, the obvious use might seem to be to select the automated mode allowing the optimizers complete freedom to vary all the variables of interest. Even if this were desirable, it is not practicable for a number of reasons. First, a single loop through the design routines requires some 90 seconds on the machines being used (complete sets of offsets being generated and analysed during each pass). Therefore, runs of a thousand loops require around 25 hours to carry out and such numbers of loops are commonly required when optimizing over many dimensional spaces. Secondly, it is necessary to verify that the optimum found by an optimizer is the true global optimum, some problems have local sub-optimal 'peaks' and others constraints that can 'stall' optimizers. Finally, it has been found that not all of the optimization methods available are suitable for the ship concept design

process. Consequently, a more sophisticated strategy is required to get the best from the system described here:-

(1) First, select an objective function, i.e., a parameter to be minimized, that is considered a reasonable measure of merit for the proposed design. This is quite likely to contain terms representing numerous desirable characteristics in the final ship.

(2) Next select as many constraints on a realistic and workable ship as can be found (stability, strength, capacity, etc.).

(3) The constraints are then tested by supplying an initial ship (normally an educated guess or a default ship) using the manual mode. If too many constraints have been selected or they have been drawn too tightly this will prove difficult (or perhaps impossible !) and this may lead to a revision of these quantities. It also allows the designer to carry out an interactive design session in the classical manner.

(4) Next choose a set of design variables that are thought to have a significant influence on the objective function, to form the trial vector. The number of variables chosen determines the dimension of the function to be optimized and since it is desirable, during preliminary work, to produce a 3-D contour mapping of the function, the list of variables is next reduced to two. These are chosen to be the most dominant ones for the problem, usually by trial-and-error. This reduction allows the generation of a contour map of the objective function for the selected variables, subject to the given constraints. This map shows the type of function involved, indicating its behaviour as the variables change, revealing whether the parameters chosen do in fact control the objective function and also highlighting which constraints are active. It also allows subsequent optimization studies to be kept away from testing unpredictable combinations of variables that might otherwise cause difficulties.

(5) For simplicity, optimize for just these two variables, so that the optimization process can be plotted on the contour map and different optimization strategies compared for the objective function under examination. During this process the system allows the user to monitor the optimization by displaying the status of the process at pre-defined numbers of cycles, or intermittently using an interrupt facility. When the various optimizers finish a status message is generated and if a true optimum has not been found, an explanation given. The reasons for failure vary; they can arise because the method chosen is not capable of further minimisation, or perhaps is unable to cope with an infeasible starting point, or simply that the amount of computer time specified by the user was too little to allow convergence.

(6) Finally, once this process has been fully studied, proceed to the n-dimensional problem that arises when carrying out the full optimization, and for which maps cannot be produced.

At this point some confidence in the final answer may be expressed without resort to exhaustive testing. However, further manual *manipulation of the design* may be informative for the user, either in verifying the qualities of the suggested design or in re-defining the optimization problem, whereupon the above strategy can be re-entered. As will be mentioned later, failures during optimization can also arise when very wide limit settings are given for the trial vector, that allow the optimizer to reach unpredicted singularities. These are usually caused by the optimizer selecting inconsistent hull-form parameters, such as extreme flare combined with a full midships section. For these reasons, monitoring and interaction is very important *during* this combined manual and automatic design process, because suitable direction to the design process can then be given. Usually, interleaved sequences of interactions and *optimizations* are found most appropriate when developing a design.

2.2. Optimizer (OPTCTL)

A number of different optimization strategies are available within the system, mostly drawn from the OPTIVAR package, which is a public domain library of general purpose optimization routines collated by Siddall[18]. When running OPTIONS in its automatic mode, a linking routine provides data from the database to OPTIVAR in a suitable form. The suite contains seven main methods of unconstrained nonlinear optimization with four associated penalty functions, together with two constrained nonlinear methods. In the jargon of optimization the measure of merit to be minimized or maximized is usually called the *objective function*, the restrictions on parameters a *constraint vector*, while the group of variables that characterize a particular solution is commonly known as a *trial vector*. It is the trial vector that is changed at the beginning of each step of an optimization search. Within the OPTIVAR package the optimizers all try to *minimize* a *single* parameter defined to be the objective function. This can be resistance for example, or any parameter or combination of parameters where minimization is desirable (most functions can be specified in this way by suitable transformations). To achieve an optimum the optimizer modifies variables selected by the user to form the trial vector (e.g., C_P, breadth to draught ratio, length, volume, etc.), according to the strategy in use. These variations are carried out within pre-established upper and lower limits set in the design data-base. When seeking constrained optima, which is the normal case, the optimizer additionally tries to fulfill separate constraint requirements (on parameters such as the enclosed volume, stability criteria, etc.,) and keeps records of constraint violations.

One additional feature that has been incorporated in the optimizer package concerns the evaluation of the objective function for values of the trial vector where the calculation breaks down (possibly due to singularities within the function, geometrical inconsistencies, etc.) When this occurs the system uses the previously calculated objective function value arbitrarily incremented by 10^{10}. Constraint functions that cannot be calculated are decremented by 10^{10}. These features tends to drive the optimizers away from areas where the system is unable to calculate data, allowing the optimization to continue where it would otherwise 'crash'. However, as is noted later, even this technique is not foolproof in cases where a particular optimizer relies on consistent data over the entire region being studied.

2.3. Design Theory Modules

The Design Theory Modules used here form a basic set of naval architectural routines. They are just sufficient to illustrate the approach although they could easily be expanded. They currently comprise:-

HULLCR for hull-form creation.
HULLDR for hull-form drawing.
STABCV for hydrostatics, condition draughts and trims and GZ curve generation.
SPACE for enclosed volume estimation.
WEIGHT for condition displacement and CG estimation.
RESHLT for resistance calculation using Holtrop and Mannen's[19,20] power prediction method.

STCIMO for stability evaluation using the I.M.O. A-287 criteria (the criteria are made more severe for the frigate example studied here).

2.3.1. Hull-form Creation (HULLCR) This routine, derived from the method developed by Keane[21], defines a simple hull-form from a set of nineteen parameters. The approach used for the below water form being a variation on the 'Lewis Form' method allowing for flare and rise of floor using very few section parameters. The above water form is given as a quadratic function, tangential to the below water form at the water-line. The water-planes and upper deck are defined as cubic polynomials in longitudinal position forward of a parallel section and quadratic aft of it. This routine produces the offsets used in the design process and suitable messages are generated if an inconsistent set of hull-form parameters has been specified.

The advantage of using this method is that hull forms can be defined extremely rapidly, in full, from few parameters, discarding complex hull fairing processes. This method of definition for sections, water-planes and profile allows great flexibility in modification, scaling and distortion of the form. These capabilities are fundamental to an optimization system, where a great many loops through a design may be required, ruling out manual hull-form manipulation methods. Of course, the method has limitations when dealing with certain types of ships, (i.e., chined ones and some extreme forms), but most types can be handled. It has been found that it is ideal for the concept design phase where great precision in representing minor form details is not warranted. To define a hull-form, the following nineteen parameters are required:-

(1) Waterline Length (L_{WL})
(2) Waterline Beam (B_{WL})
(3) Draught (T)
(4) Depth (D)
(5) Block Coefficient (C_B)
(6) Maximum Section Coefficient (C_X)
(7) Overall Beam at the position of Maximum Section (B_{OA})
(8) Length of Keel (L_{KEEL})
(9) Length of the Parallel Middle Body (L_{MID})
(10) Non-dimensional Position of Maximum Beam (L'_X)
(11) Ratio of Waterline Transom Width to Waterline Beam (BT_{WL}/B_{WL})
(12) Ratio of Overall Transom Width to Overall Beam (BT_{OA}/B_{OA})
(13) Rise of Floor Angle (ROF)
(14) Waterline Flare at Maximum Beam ($FLARE_X$)
(15) Waterline Flare at the Aft Perpendicular ($FLARE_A$)
(16) Half Angle of Water-plane Entrance (i_E)
(17) Upper Deck Half Angle of Water-plane Entrance ($i_{E\,UD}$)
(18) Forward Rake Angle ($RAKE_F$)
(19) Aft Rake Angle ($RAKE_A$)

With the rule-base currently used by the design control module only ⓜ, C_P, B_{WL}/T, D/T, C_B, ∇ (displaced volume) and a ship type specifier need be input to define the hull-form. From these primary parameters L_{WL}, B_{WL}, T, D and C_X are generated and a type specifier used to indicate a set of non-dimensional defaults for the thirteen remaining parameters. If desired, the user may input all of these thirteen parameters or just some of them allowing the designer to impose a chosen style on the final hull-form. The defaults provide convenience by using type ship data to provide a useful starting point for design. Default parameters are currently available, in non-dimensionalized form, for three types of ships, i.e., passenger ships, bulk carriers, and frigates. These could easily be supplemented by previous designs produced by the user. The three actual default ships are generated if all the input data are derived from the defaults. Otherwise, the user starts with a typical ship of the chosen type, that can then be modified during optimization (note this does *not* imply that the program is restricted to simple distortions or scaling and each of the default ships can be continuously altered to reach any of the others). In practical design work it is likely that the main particulars would be given, and the non-dimensional versions derived from them. This is possible within OPTIONS, but when examining wide ranges of parameters it is found simpler to start with their non-dimensional equivalents.

2.3.2. Hull-form Drawing (HULLDR) This routine allows the hulls created by HULLCR to be plotted using the GINO package. Body-plans, water-planes and 3-D perspective views can all be generated.

2.3.3. Hydrostatics and Stability Curves (STABCV) This routine is a general purpose ship stability program that uses polar ordinates with 5° variation for 21 transverse sections, defining the hull-form using appropriate integrating multipliers and levers and allowing for the inclusion of appendages. It can calculate hydrostatics, cross curves of statical stability (at constant trim) and condition GM and GZ curves (allowing for trim induced by heeling). This module is used for three purposes; first the hydrostatics at the specified design draught and level trim are used as input by subsequent modules such as SPACE and WEIGHT. Secondly, having established various condition displacements and centres of gravity using WEIGHT, it is used to establish condition draughts and trims for use in powering calculations etc. Finally, the GZ curves are used in the application of the stability criteria, usually as constraints during the design process.

2.3.4. Enclosed Volume (SPACE) This routine is still just the embryo of a 'volumes and areas' routine for concept design. Currently, it only calculates the enclosed hull volume from the hydrostatic data and a given (or default) superstructure size. It is used to ensure the availability of sufficient internal space for the design mission (deadweight, payload, etc.). It is currently crude but adequate for the purpose, ensuring against designs with insufficient payload capacity. Further developments will address the positioning of decks, bulkheads and machinery spaces.

2.3.5. Condition Displacement and CG Estimates (WEIGHT) This routine estimates weights and vertical centres of gravity, carrying out assessments for the number of condition displacements desired by the user. At present, only passenger ship, bulk carrier and frigate outfit data are available (e.g., see Watson and Gilfillan's[22] work for the hull, outfit and machinery weights of commercial ships). The vertical centre of gravity estimates are derived using regression techniques on existing ship data. Notice that, as is common in commercial practice, the hull design condition is for a draught and trim that may, or may not, coincide with a particular operating condition. This differs from normal warship practice where the hull-form is designed at the deep displacement which is also the principal operating condition. This method can be adopted here by setting the hull design displacement to be equal to a calculated condition displacement at the end of every loop of the design procedure. However, such an approach tends to cause upwards spirals in displacement and

is not necessary when a detailed hull-form is available allowing several conditions to be analysed with equal accuracy. If some requirement on length, displacement draught, etc., is to be met for a given condition this is more simply handled as a constraint during the optimization process.

2.3.6. Resistance Calculation (RESHLT) This routine is based on Holtrop and Mannen's power prediction method[19,20] and is fully in accordance with that method, except that it currently does not include the propulsion analysis (instead a designer chosen propulsive coefficient is used). The method calculates the total resistance from the addition of the following components, ignoring any interaction between them, i.e.

$$R_T = (1+K_1)R_F + R_{APP} + R_W + R_B + R_{TR} + R_A$$

where R_F is the frictional (ITTC) resistance, $(1+K_1)$ the form factor for the hull, R_{APP} the appendages resistance, R_W the wave resistance, R_B the additional pressure resistance of a bulbous bow near the surface, R_{TR} the addition pressure resistance due to transom immersion and R_A the model ship correlation allowance. Appendages, bulbous bows, bow thrusters and different stem shapes can be specified by the user or default values used instead.

The choice of Holtrop and Mannen's method is based on the fact that it is acceptably accurate for concept design purposes and covers a wide range of types and sizes of ships. Possible singularities that may arise in this method due to inconsistent or extreme combinations of C_P and longitudinal centre of buoyancy are rejected by the routine HULLCR, since they represent impossible forms.

2.3.7. Stability Criteria Verification (STCIMO) This routine was developed to check intact stability by applying relevant parts of the I.M.O. A-287 criteria. Specifically, the GZ curves and initial GM calculated by STABCV for each condition displacement are verified according to :-

(1) Initial GM ≥ 0.15 m.
(2) Area below GZ curve from 0 to 40° ≥ 0.09 m rad.
(3) Area below GZ curve from 30 to 40° ≥ 0.03 m rad.
(4) Area below GZ curve from 0 to 30° ≥ 0.055 m rad.
(5) Angle of Maximum GZ ≥ 30°.
(6) Maximum GZ ≥ 0.2 m.

Note that these criteria are intended for commercial vessels and those for warships would be much more severe. To allow for this variation, the percentage exceedences of the criteria may be used as constraints during optimization, allowing the user to require greater or lesser compliance than the I.M.O. regulations lay down.

2.4. Data-base Handler (GENDAT)

As part of this program of work, a design data-base handler has been developed in the form of a library of Fortran functions that may be readily incorporated into new programs. The primary function of this package is to provide the means of controlling and accessing the design data-base. The requirement for flexibility in this capability means that, in addition to providing the means for storing and retrieving data, it also has to provide a powerful, program definable, user interface. In common with most such interface packages, continual improvements are being made to this suite; currently it is being reworked to make use of the mouse driven X-windows system commonly used by modern workstations.

When the design control module is run, this interface is entered and it initiates a menu driven interactive process with the user. It is from this level that all subsequent routines are accessed and controlled. Using the package a ship description file can be recalled from disk, and calculated results sent back. It additionally incorporates features that allow the input of values for the various parameters of interest and the selection of suitable constraints and objectives together with the parameters to be varied during optimization. It also allows the creation of formated output files for subsequent printing. The design data-base maintained by this system contains the names, values, types, units, meanings, etc., for the various quantities making up the vessel description. These can be accessed individually or in groups, enabling the user to have great flexibility and ease of control. Finally, it is through this interface that commands are given to enter either design mode of OPTIONS.

3. EXAMPLE DESIGN

An example design has been carried out to illustrate the capabilities of this system and in particular to evaluate the embedded optimization techniques. A frigate of some 3,300 tonnes deep displacement (Δ_D) has been designed manually as a starting point for optimization, with the goal set as minimum resistance at design speed within the usual constraints (i.e., stability, strength, payload capacity, etc.). The initial hull-form, which is designed on a fixed displaced volume close to the light displacement, has a fairly high breadth to draught ratio ($B_{WL}/T = 4.0$) and a low length to volume ratio ($\textcircled{m} = 7.0$). This allows the system scope to improve the design and also to avoid pre-judging the 'best' solution. Of course, it is to be expected that rather predictable changes would be made to improve this design. However, this is a consequence of asking a question to which the answer is fairly well established, the real strength of optimizers lies in their ability to deal with many competing aspects at one time, whether or not the user is able to predict the likely consequences of attempting to satisfy such requirements, i.e., a known task has been specified to show that the *optimizers* can achieve the desired result without *a priori* knowledge.

The initial input data for the frigate considered are given in Table 1 (with a summary of the corresponding output data in column one of Table 5). The mission requirements for this design are represented by a design speed of 30 knots in the deep

$\textcircled{m}$	7.0	C_P	0.57
B_{WL}/T	4.0	∇	2400 m^3
C_B	0.47	D/T	2.5

Table 1 - Initial input data.

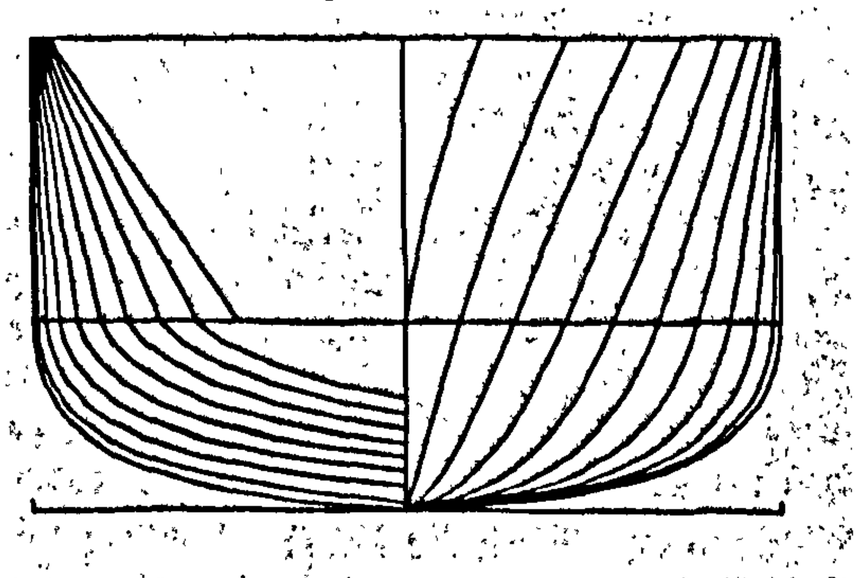

Figure 2 - Body plan for the initial design.

Optimization	with 2 variables	with 5 variables
Objective Function	Total Resistance R_T (KN)	
Constraint Vector and Limits	Initial GM ≥ 0.7 m. Area below GZ curve from 0 to 40° ≥ 0.09 m rad. Area below GZ curve from 30 to 40° ≥ 0.03 m rad. Area below GZ curve from 0 to 30° ≥ 0.055 m rad. Angle of Maximum GZ ≥ 30°. Maximum GZ ≥ 0.2 m. $L_{WL}/D \leq 14.0$ $12000 \leq \nabla_{ENC}$ $0.0 \leq L_{WL} \leq 1000.0$ $0.0 \leq B_{WL} \leq 1000.0$ $0.0 \leq T \leq 1000.0$ $0.0 \leq D \leq 1000.0$ $0.4 \leq C_X \leq 1.0$	
Trial Vector and Limits	$5.0 \leq$ ⓜ ≤ 12.0 $1.0 \leq B_{WL}/T \leq 6.0$ $C_P = 0.57$ $FLARE'_X = 0.0$ $L'_X = 0.52$	$5.0 \leq$ ⓜ ≤ 12.0 $1.0 \leq B_{WL}/T \leq 6.0$ $0.55 \leq C_P \leq 0.60$ $0.0 \leq FLARE'_X \leq 10.5$ $0.14 \leq L'_X \leq 0.60$
Fixed Parameters	$\nabla = 2400 m^3$ $C_B = 0.47$ $D/T = 2.5$	

Table 2 - Optimization Conditions (Summary).

condition, together with a payload consisting of a fuel deadweight of 600 t at 0.8 m above the keel plus equipment totaling 175 t at a height which varies according to the hull and superstructure particulars. The fuel load is held fixed for simplicity, i.e. the maximum cruise range will vary, also the performance calculations are only to be carried out at full load displacement. Notice that the final deep displacement is *not* specified explicitly, since this is derived during the analysis of the hull, but instead the displaced volume has been given for the hull design (here, light) condition, also these two differ substantially. This design meets all the criteria subsequently used as constraints during the optimization studies and a body plan is given as Figure 2.

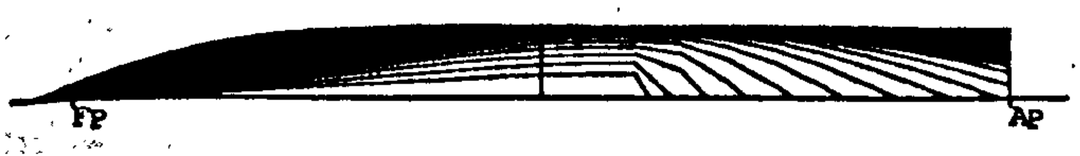

Figure 3a - Variation of non-dimensional position of maximum beam, L'_X (aft).

Figure 3b - Variation of non-dimensional position of maximum beam, L'_X (forward).

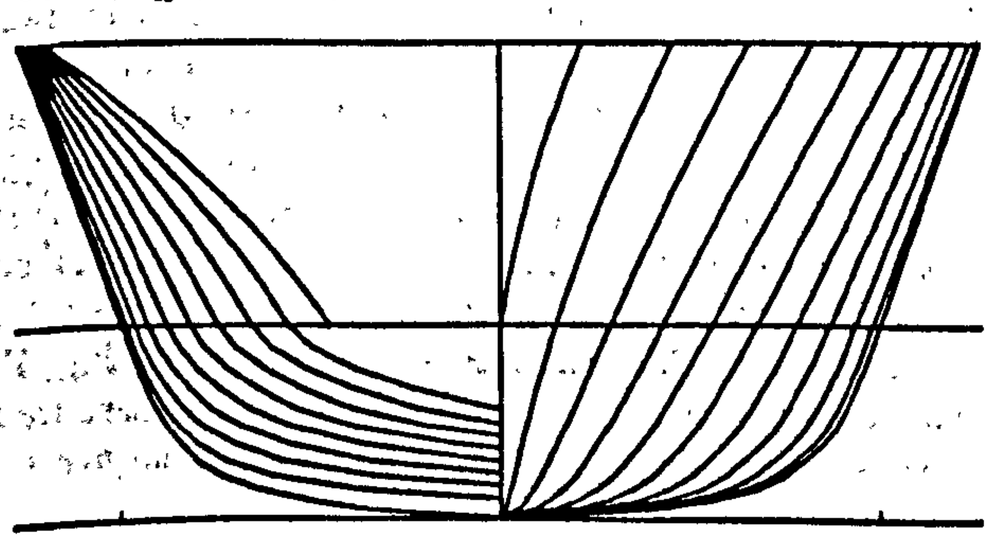

Figure 4 - Variation of waterline flare at max. beam, $FLARE_X$.

3.1. Selection of Objective Function, Constraints and Variables

The objective function, constraints and design variables selected for this problem are summarized in Table 2. As has already been mentioned, the objective function selected for minimization was the total resistance at design speed. The constraints adopted were the six stability criteria (notice that a non-standard, rather severe level has been specified for the initial GM criterion), an upper limit on the length to depth ratio (L_{WL}/D) to ensure against longitudinal strength problems and a minimum enclosed volume (∇_{ENC}) to guarantee payload capacity. Additionally L_{WL}, B_{WL}, T, D and C_X were constrained, but within very wide limits. The design variables chosen were those that would strongly affect both resistance and stability. These were ⓜ, B_{WL}/T, C_P, $FLARE'_X$ (non-dimensional $FLARE_X$) and L'_X. Limits were chosen for these variables by making test runs, which established that values outside the given ranges would produce geometrically inconsistent ships. The system is able to cope with such inconsistencies, but these settings help it to run faster since it does not need to examine clearly unworkable combinations. In fact, some extreme combinations of variables, such as $C_P = 0.56$ and $L'_X = 0.3$, for ⓜ = 7.0, $B_{WL}/T = 4.0$ and $FLARE'_X = 0.0$, still produce inconsistent ships, and these cases were left in to exercise the system's ability to overcome such difficulties. Figure 3 illustrates the range of influence of L'_X on the hull-form, while Figure 4 shows that for extreme $FLARE_X$.

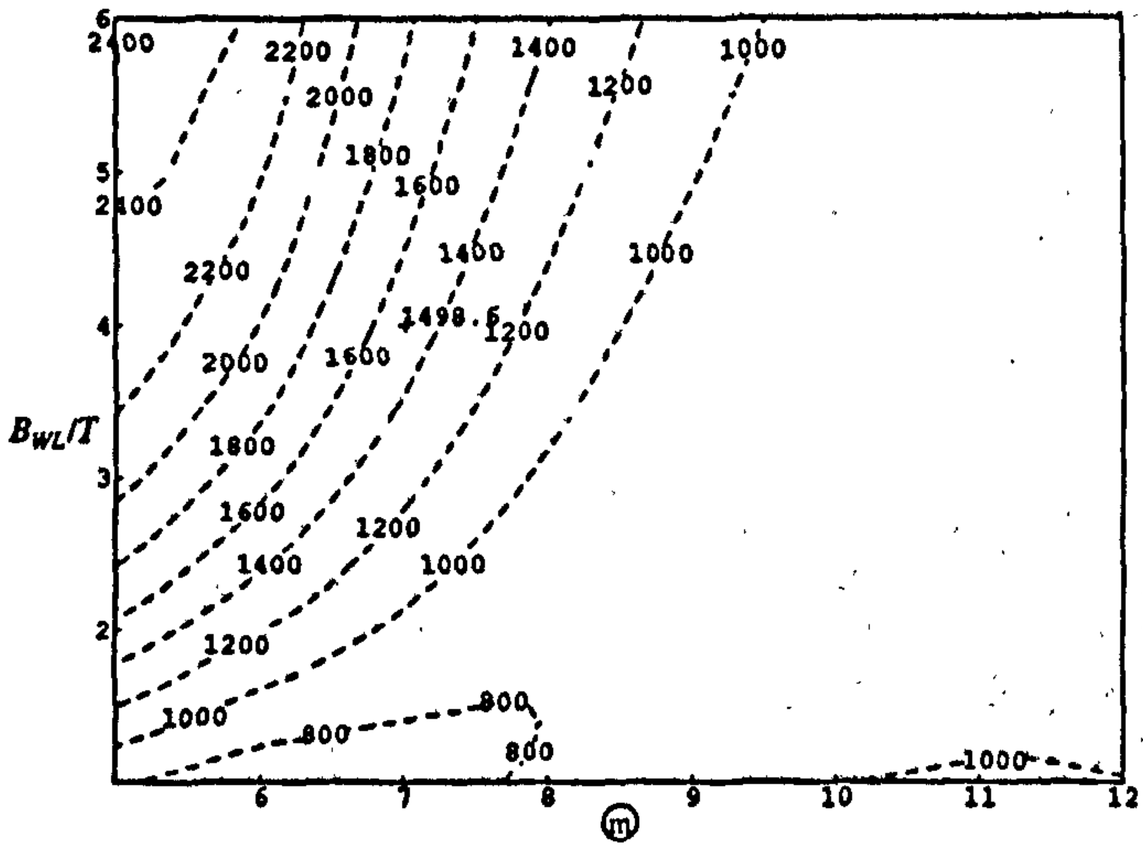

Figure 5a - Contour map of total resistance, R_T (KN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T.

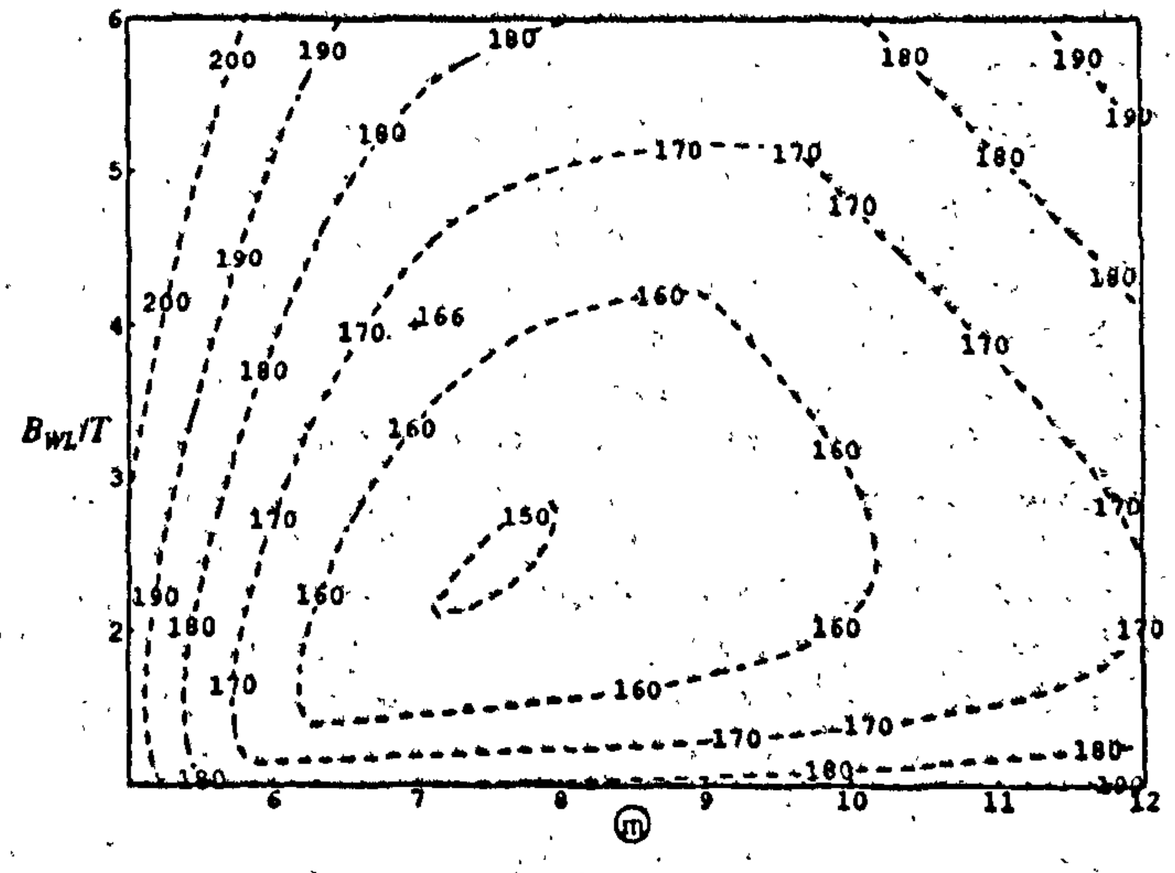

Figure 6a - Contour map of total resistance, R_T (KN) at 15 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T.

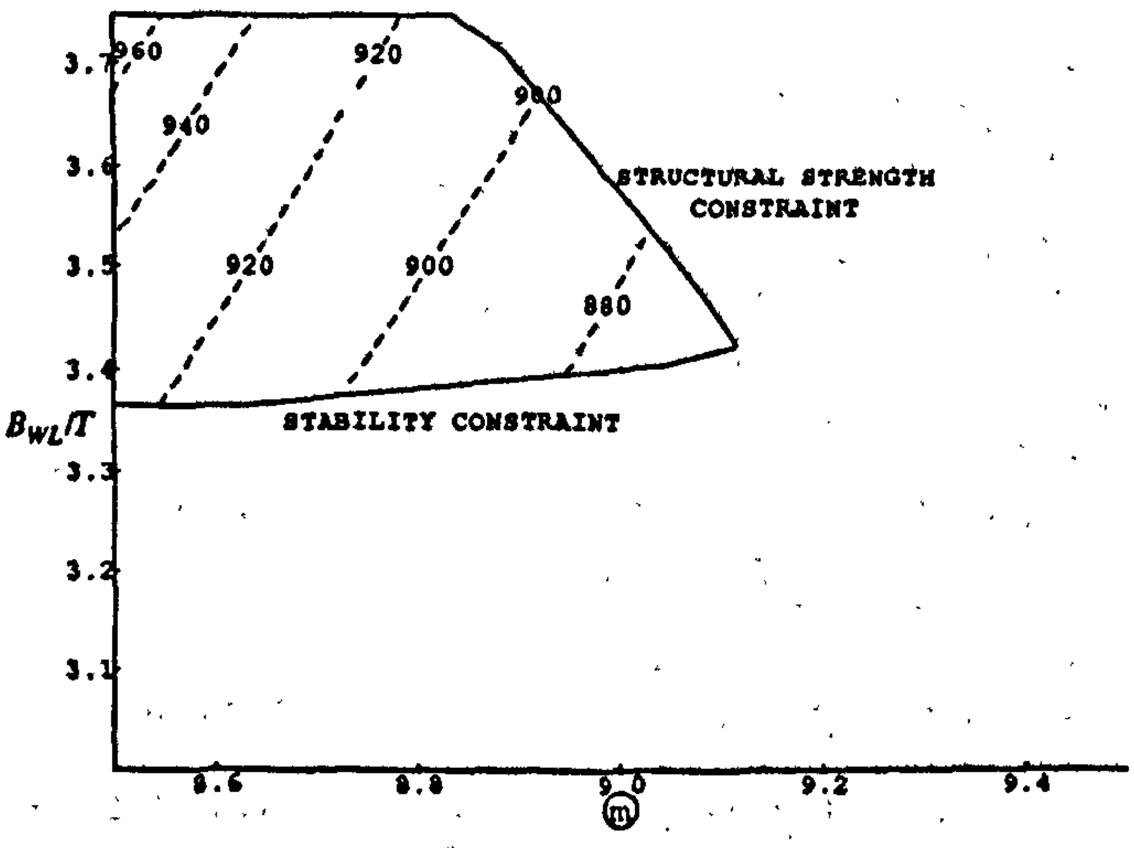

Figure 5b - Detail contour map of total resistance, R_T (KN) at 30 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T (taken from Figure 5a but with constraint boundaries shown).

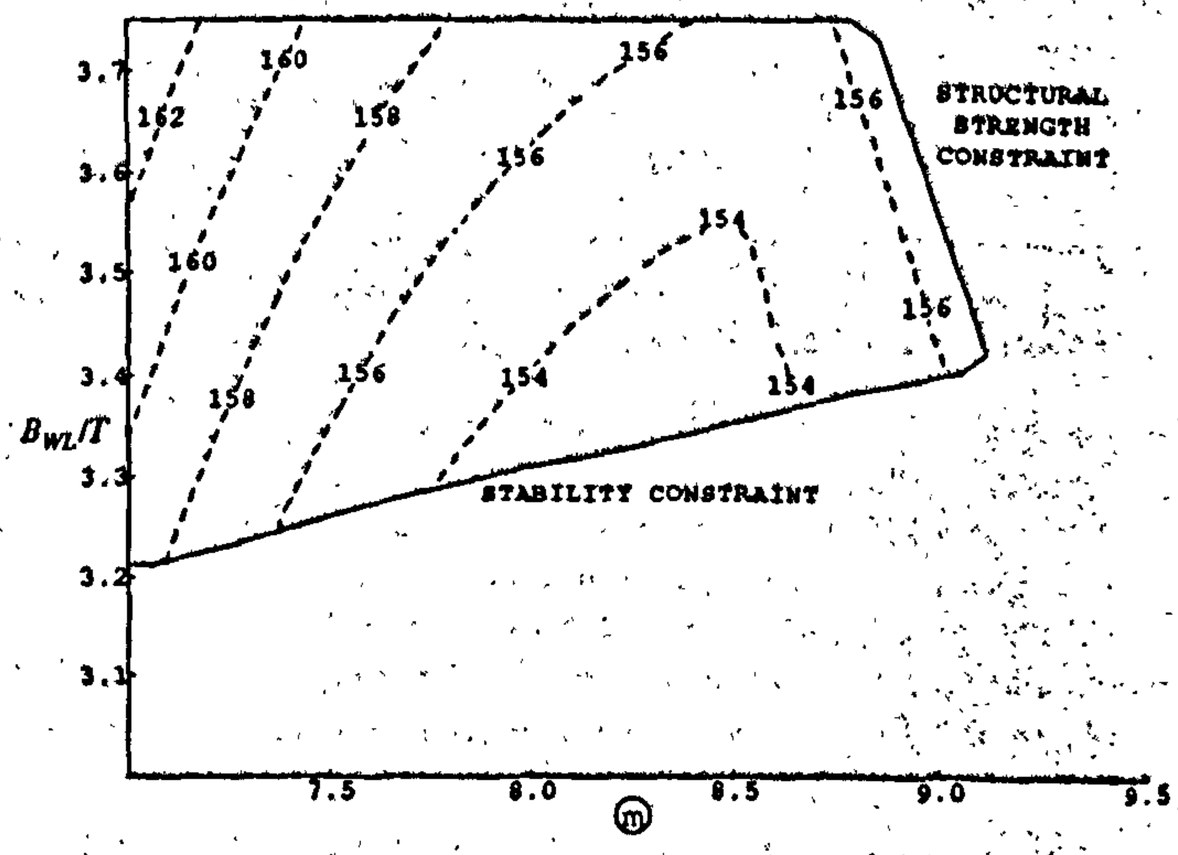

Figure 6b - Detail contour map of total resistance, R_T (KN) at 15 kts. for length to displaced volume ratio, ⓜ versus waterline beam to draught ratio, B_{WL}/T (taken from Figure 6a but with constraint boundaries shown).

3.2. Two Dimensional Mapping of the Objective Function

In order to enable the objective function (R_T) to be mapped, initially only ⓜ and B_{WL}/T were allowed to vary, the other variables remaining fixed as originally given i.e., $C_P = 0.57$, $L'_X = 0.52$ and $FLARE'_X = 0$. This mapping is shown in Figure 5 where it can be seen that resistance does not necessarily decrease continuously as the ship gets more slender, due to increase of the wetted area. When constrained by stability and structural strength, the compromise optimum lies in a region with B_{WL}/T a little greater than 3.4 and ⓜ at approximately 9.1, and this is entirely as expected for this reduced problem, being in line with the traditional practice of high speed frigate design. A sensitivity exercise is shown in Figure 6, where the ship is optimized at a different speed, here 15 knots. In this case, an unconstrained optimum is obtained at ⓜ ≈7.6 and B_{WL}/T≈2.5, indicating how speed sensitive such designs are. The detail mapping shows the previous constraint boundaries, which are unchanged with this different objective function. Here, only the stability constraint is active, indicating that a constrained optimum occurs at ⓜ ≈8.2 and B_{WL}/T≈3.3,

Although outside the scope of the strategy previously outlined, it is perhaps instructive to map the other three variables of interest with ⓜ fixed at 7.0 and B_{WL}/T at 4.0, see Figures 7 and 8. The variables C_P, L'_X and $FLARE'_X$ produce profound changes to the ship, and because of this can only be varied over rather limited ranges if feasible designs are to result. In Figure 7, it should be noted that the boundaries are not explicit constraints, but rather the limit for geometrically consistent ships. In Figure 8, the left-hand boundary is caused by the application of the stability criteria, whilst the upper one is again the limit for consistent ships. The sloping left-hand boundary of Figure 8 indicates that increased flare increases stability, allowing B_{WL}/T to be reduced and hence resistance decreased. Again, these trends might be as expected, but the *best combination* of all *five variables* is less apparent.

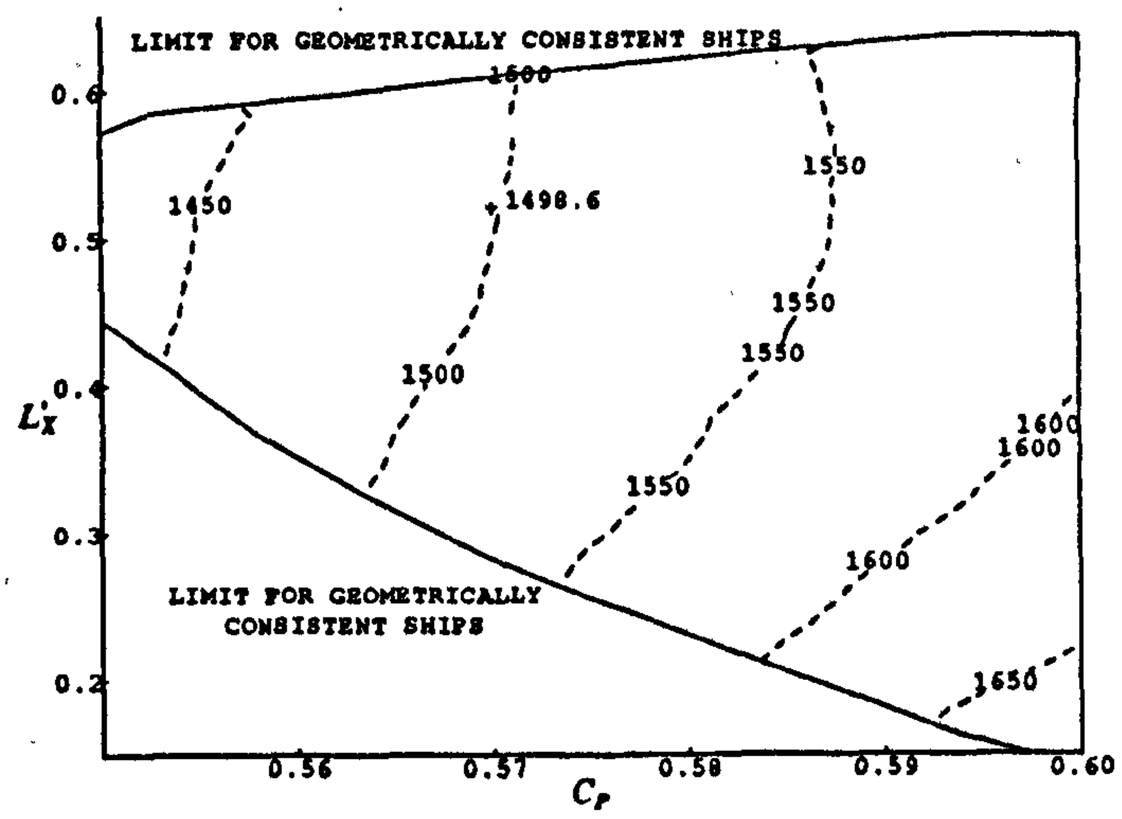

Figure 7 - Contour map of total resistance, R_T (KN) at 30 kts. for prismatic coefficient, C_P versus non-dimensional position of maximum beam, L'_X.

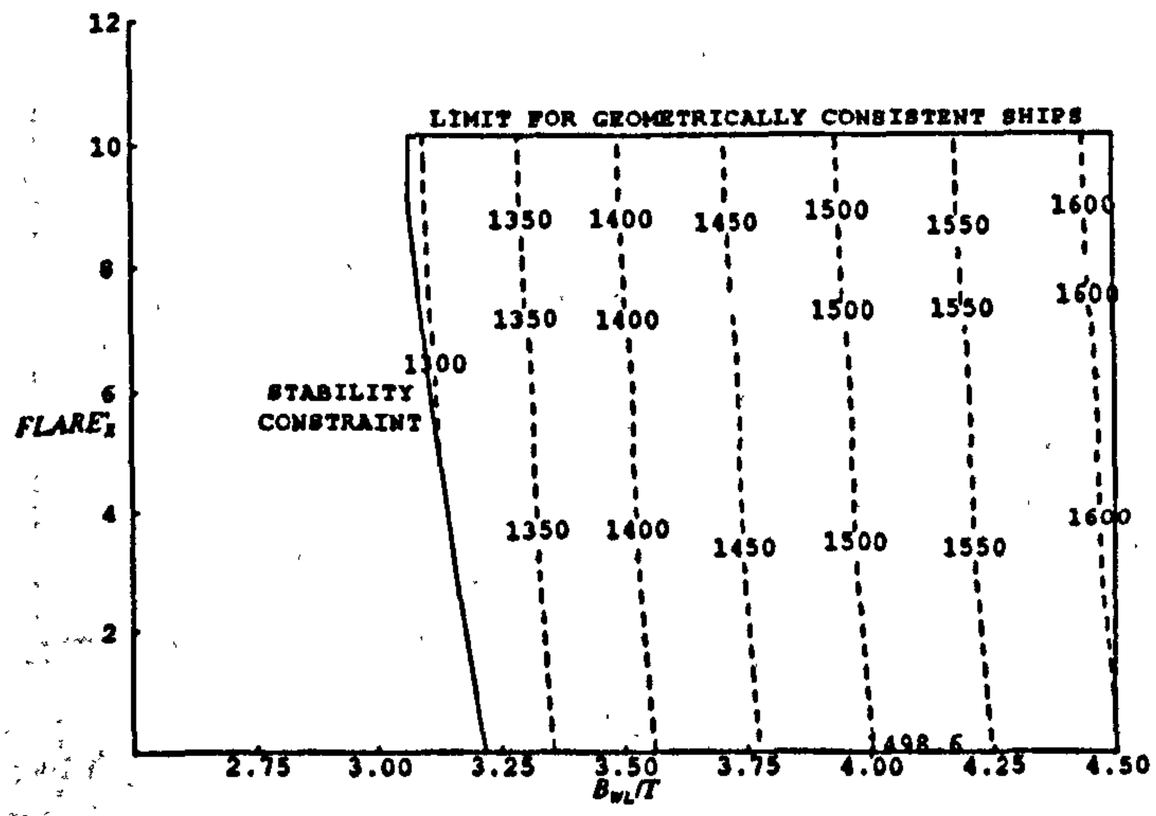

Figure 8 - Contour map of total resistance, R_T (KN) at 30 kts. for waterline flare at maximum beam, $FLARE'_X$ versus waterline beam to draught ratio, B_{WL}/T.

3.3. Optimization with Two Variables

Continuing with the previous strategy, the various optimizers are next applied to the two variable problem, varying ⓜ and B_{WL}/T with C_P, $FLARE'_X$ and L'_X fixed. A brief outline of the optimization methods that have been found useful in the present context is given in the Appendix. Although this material is widely known to those familiar with optimization it is repeated here since it may be of interest to a wider readership. The methods used were:-

(1) Successive linear approximation - APPROX;

(2) Random exploration with shrinkage - RANDOM;

(3) Hooke and Jeeves direct search - SEEK, with two types of penalty functions:-

a) one pass external function - OPTIM1;

b) Fiacco-McCormick combined external and internal function - OPTIM2.

Each of the methods used has a number of control parameters to set step sizes, number of loops, tolerances, etc., all of which may be selected as required. With two variables, the maximum number of design loops was set at 1000 and all other features taken as default.

Given the previous contour maps of this problem the optimum is known and the purpose here is to establish that the optimizers could, in fact, reach this sub-goal. The results achieved are summarized in Table 3 and are next discussed in turn :-

APPROX

This method gave a successful optimization, the minimum being reached very rapidly, in 24 loops (or ship designs). The optimization path is shown in Figure 9a and it can be seen that the expected minimum was reached (c.f., Figure 5b).

RANDOM

With the maximum number of loops set to 1000, this routine fails to reach the expected minimum although it indicated that it was satisfied with the result achieved. In fact the optimizer stopped after 123 assessments of the objective function (hull-form and resistance), which were preceded by 185 assessments of the constraints (hull-form and stability).

Method		APPROX	RANDOM	SEEK	SEEK	RANDOM *
Penalty Function		n/a	n/a	OPTIM1	OPTIM2	n/a
Success/Failure		Success	Success	Success	Success	Success
Number of Loops		24	185 †	369	613	503 †
Objective Function	R_T (KN)	855.0	1064.9	855.1	855.1	876.8
Trial Vector	ⓜ	9.143	7.974	9.143	9.143	8.906
	B_{WL}/T	3.410	3.542	3.410	3.411	3.399
Constraint Vector	Initial GM (m)	0.70	1.06	0.70	0.70	0.71
	Area 0-40° (m rad)	0.23	0.31	0.23	0.23	0.23
	Area 30-40° (m rad)	0.11	0.14	0.11	0.11	0.11
	Area 0-30° (m rad)	0.12	0.16	0.12	0.12	0.12
	Angle Max GZ (°)	54	53	54	54	54
	Max GZ (m)	0.92	1.14	0.92	0.92	0.92
	L_{WL}/D	14.0	11.6	14.0	14.0	13.4
	$\nabla_{ENC}(m^3)$	13588	13698	13588	13588	13582

Table 3 - Optimization with Two Variables.
* modified shrinkage action, see text. † objective function evaluated at feasible combinations only.

SEEK a) Using a one pass external penalty function (OPTIM1) this method successfully reached the optimum. When constraint lines were violated (at first stability and subsequently L_{WL}/D), the penalty function successfully drove the variables back to the feasible area. After 90 loops, 100 random points were tried in the vicinity of the last result, and the best result used as a new starting point. After 369 further loops, including a second set of 100 random designs, it finally reached the minimum. These results are very similar to those generated by APPROX. The path followed, including the shotgun searches, can be seen in Figure 9b.

b) Using the Fiacco-McCormick combined external and internal penalty function (OPTIM2) this method again gave a successful optimization, using the same strategy as with OPTIM1, but with a smoother penalty function. When using this penalty function there is no random search, as it is designed to avoid working far from the feasible region. The process ended after 613 loops, requiring more steps because of the repeated sub-optimizations used. The results are very close to those given by SEEK with OPTIM1 and also APPROX. The general path of the optimization process is shown in Figure 9c.

The only problem encountered with these two dimensional optimizations concerns the failure of RANDOM to reach the true optimum, especially since this routine ought to provide a standard for comparison. This failure can be directly attributed to the shrinkage mechanism, combined with the small number of points tested between each reduction in the search area. This mechanism causes the random search to be concentrated in areas found to be giving good results, allowing the true optimum to be rejected under some circumstances. This arises because unfeasible points are rejected, irrespective of the objective function value at the location tested; consequently this method tends to miss optima that are defined by constraint boundaries, as here. Nonetheless, it should be noted that the routine still produced a reduction in resistance of some 30% compared to the original design. To overcome this difficulty, and to increase confidence in the method, the shrinkage mechanism must be made less severe. By default, it takes ten times as many points as variables at each stage, of which it keeps the best 25% (i.e., five from twenty for this two dimensional problem) and this set of best points is used to construct the next shrunken range for investigation. Increasing the total sample to eighty whilst still retaining the best five, yields the desired result, although the exact precision of the other methods is still not achieved, see again Table 3.

3.4. Optimization with Five Variables

Having established under what conditions the optimizers work with two variables and also the likely effects of modifying the different variables under investigation, it is possible to move on to the full problem, i.e. a five parameter optimization. Although this multi-dimensional process cannot be mapped, it is to be expected that the best ship would lie in a region where Ⓜ is about the same as for the two dimensional problem. B_{WL}/T might be slightly smaller here combined with increased flare to restore stability. Also a minimum prismatic coefficient is probable, combined with a position of maximum beam set well aft (this shifts the longitudinal centre of buoyancy aft and also tends to produce a smaller angle of entrance for the water-plane). These last three variables produce profound changes in the hull-form, compared to the simple scaling caused by altering Ⓜ and

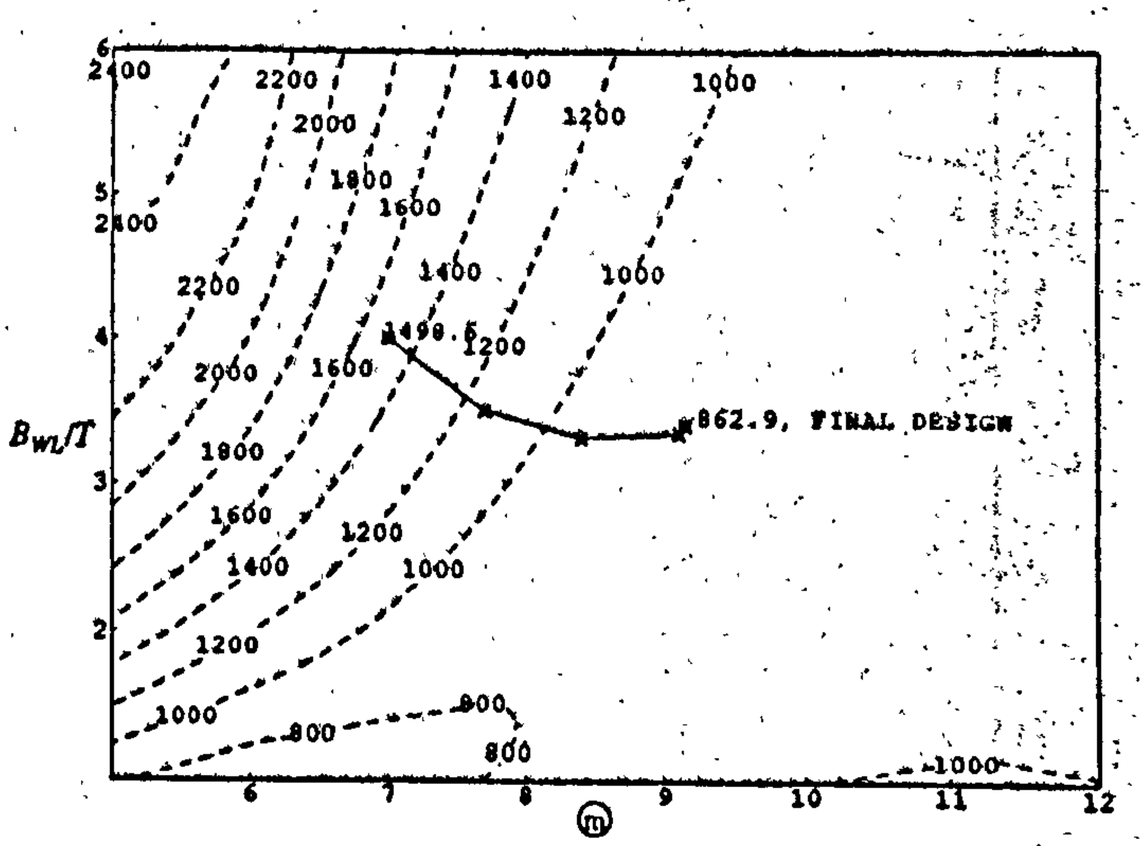

Figure 9a - Optimization path for APPROX with two variables.

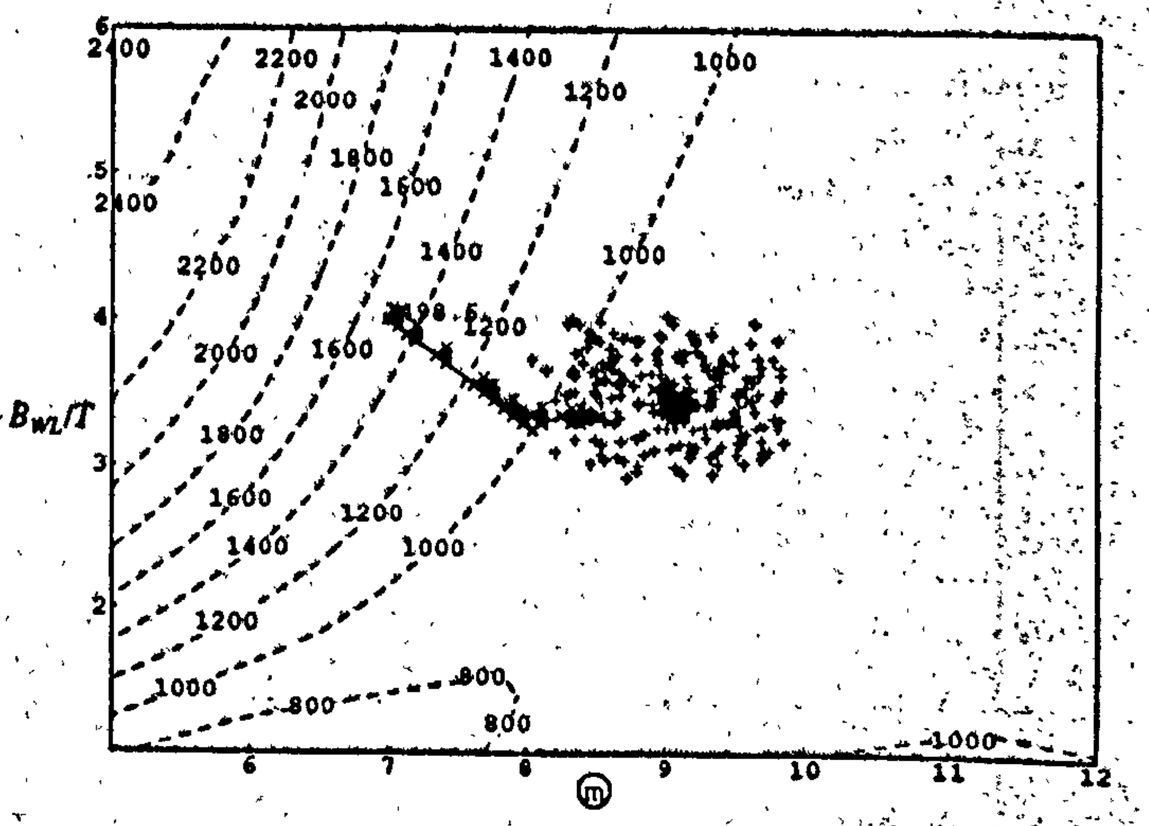

Figure 9b - Optimization path for SEEK and OPTIM1 with two variables. (Points marked '+' are produced by the shotgun searches - see appendix.)

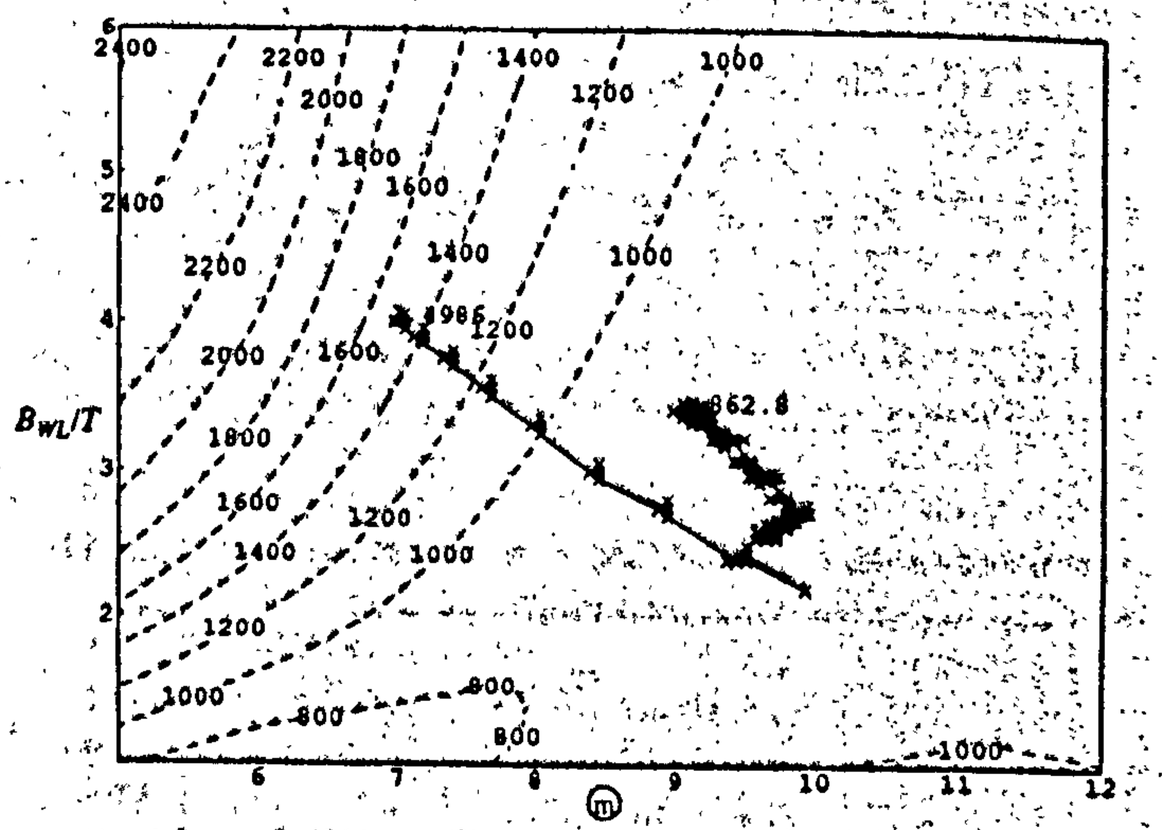

Figure 9c - Optimization path for SEEK and OPTIM2 with two variables.

B_{WL}/T. Therefore, the optimization becomes heavily constrained because of the large number of geometrically inconsistent hull-

forms that can arise. This aspect tends to produce a narrow corridor of feasible designs through the multi-dimensional space, which may be hard for the optimizers to enter or stay within. Clearly the five dimensional problem is considerably more taxing for the method; the results achieved by the various optimizers are given in Table 4 and are again discussed in turn :-

APPROX

This method fails after only six loops through the design process. It is completely unable to cope with geometrically impossible ships for which correct information cannot be given concerning the objective function. As has already been mentioned, the system attempts to overcome such impossible combinations of parameters by synthesizing values for the objective function and constraints. When only one impossible design is encountered whilst linearizing the objective function this just produces a massive distortion directing the search away from the trouble spot; this happened occasionally during the two dimensional search. However, when several such combinations occur, as happens with the five dimensional search, the resulting hyper-plane no longer points back towards sensible designs, and moreover loses all similarity to a smooth surface when the next linearized region is being constructed. The search gets hoplessly confused and gives up. It is difficult to see a way around this dilemma, this being a fundamental shortcoming of this otherwise powerful technique when applied to highly constrained problems.

RANDOM

RANDOM suffers from no such drawbacks, since it is unconcerned with the relationships between the various points tested, merely keeping a best subset. This method was applied allowing up to 5,000 combinations of the five variables, since it was found that it needed to try a very large number of combinations to get near to the minimum, many combinations being rejected due to infeasibility. As can be seen from Table 4, the search did not find the true minimum resistance with the default shrinkage mechanism. As has already been noted, if the random search does not hit a point in the vicinity of the best results at least once, the shrinkage technique used to improve speed tends to throw away the whole region, concentrating the search in the area where the best feasible point was found. The general reliability of a pure random search is thus jeopardized when the shrinkage technique is introduced to make it work more quickly. Consequently, when the function being optimized is such that one cannot predict in which region of the n-dimensional space the optimum lies, the shrinking technique should be reduced in severity. The resulting runs therefore tend to be extremely long, but the optimum can always be found. By default, with five variables RANDOM takes fifty samples and keeps the best thirteen. Increasing the total to 200 and keeping the best sixteen causes the method to examine very many more combinations, but in the end it *does* achieve the desired result (see again Table 4).

SEEK

a) Using a one pass external penalty function (OPTIM1) this method succeeded after 539 designs. The direction of search being decided, as before, by trying changes to the variables, one at a time. This approach produces a consistent path leading towards the optimum. When inconsistent hull-forms were encountered the modified objective function already discussed drove the search successfully back on course.

b) Using the Fiacco-McCormick combined external and internal penalty function (OPTIM2) the method failed to find a satisfactory optimum with five variables. However, the method did produce the design with least resistance, but with failure indicated due to the violated limit of the C_P component of the trial vector. The fact that this

Method		APPROX	RANDOM	SEEK	SEEK	RANDOM *
Penalty Function		n/a	n/a	OPTIM1	OPTIM2	n/a
Success/Failure		Failure	Success	Success	Failure	Success
Number of Loops		6	1395 †	539	1926	6336 †
Objective Function	R_T (KN)	-	923.6	832.0	829.2 $	835.0
Trial Vector	ⓜ		8.370	9.149	9.148	9.220
	B_{WL}/T		3.314	3.396	3.404	3.303
	C_P	-	0.562	0.550	0.547	0.558
	$FLARE'_X$		1.5	0.4	0.0	4.7
	L'_X		0.483	0.581	0.580	0.566
Constraint Vector	Initial GM (m)		0.72	0.70	0.70	0.70
	Area 0-40° (m rad)		0.24	0.23	0.23	0.25
	Area 30-40° (m rad)		0.12	0.12	0.12	0.13
	Area 0-30° (m rad)	-	0.12	0.12	0.12	0.12
	Angle Max GZ (°)		55	54	53	54
	Max GZ (m)		1.04	0.97	0.96	0.98
	L_{WL}/D		12.1	14.0	14.0	14.0
	$\nabla_{ENC}(m^3)$		13820	13647	13580	14425

Table 4 - Optimization with Five Variables.

* modified shrinkage action, see text. † objective function evaluated at feasible combinations only.

$ last result with sufficient stability before failure.

violation is not great indicates that with further iterations the process would probably converge, although this is not carried through here.

As predicted, the various optimization processes led to approximately the same optimum design, according to the restrictions imposed by the various constraints and the choice of variables. The process tends to produce a longer and more slender ship, increasing the length and decreasing the beam up to a certain extent. The limit reached is very much one of compromise between reducing wave making resistance and loss of strength and/or stability. The most rapid optimization, within the specified constraints, was achieved for five variables by the Hooke and Jeeves direct search with a one pass external penalty function. This final design is illustrated in Figure 10. The improvement in resistance, from the original starting point, is of the order of 81% when varying ⓜ and B_{WL}/T only, with a further decrease of 2.8% being achieved when C_P, the position of maximum beam and flare were varied. Notice also, that this is achieved despite the increase in deep displacement of some 200 tonnes caused by the changes in hull dimensions. Again, it must be emphasized that these rather predictable results arise because a well known problem has been examined, the aim being to demonstrate the technique in a realistic setting. The details of the final designs achieved using all the methods described here are given in Table 5, along with those for the original design.

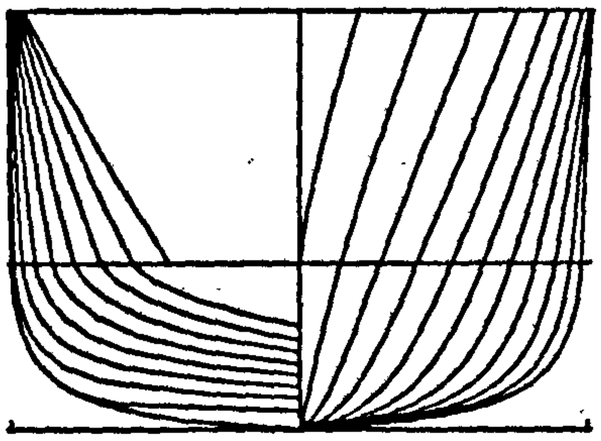

Figure 10a - Body plan for the best, constrained design.

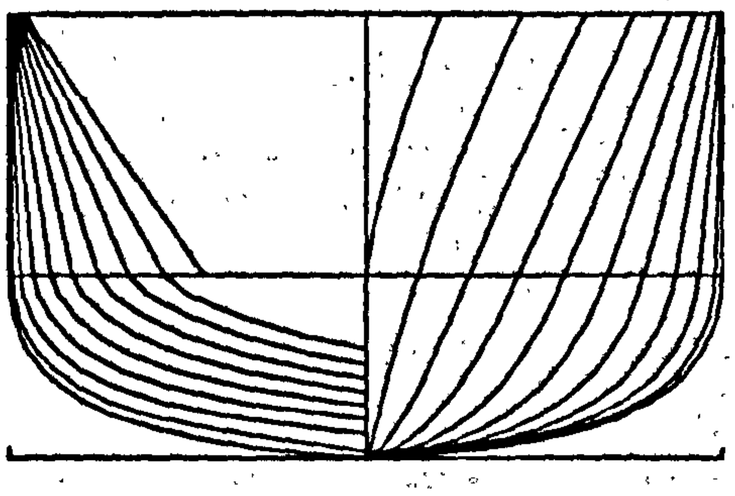

Figure 10b - Body plan for the initial design.

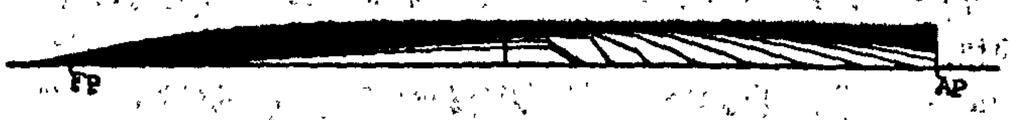

Figure 10c - Water-planes for the best, constrained design.

Figure 10d - Water-planes for the initial design.

4. CONCLUSIONS

This paper has described a ship concept design system which makes heavy use of optimization techniques, but which also draws on established theory and rule-based structures. It has been illustrated by its application to a simple frigate design problem. Fundamental to this work has been the adoption of existing naval architectural theory to carry out *all* the necessary design calculations. Such modules of theory can be chosen at will to suit the design being made; if a number of possible modules are available their effects can be examined in a systematic fashion. This allows the designer to control the design process and also to verify that the optimizer suggested designs are acceptable when tested against established practice. The fundamental idea being discussed here concerns the way a traditional design spiral may be adapted for use with modern computational methods. The approach taken here employs existing optimization techniques *and* a powerful hull definition method to automate the synthesis *as well as* the analysis of competing designs. The ability to radically modify the chosen hull-form would appear crucial to the use of optimization during this phase of ship design.

The example chosen illustrates that optimization based concept exploration is fully capable of identifying the best design, given a suitable objective, constraints and a set of variables to manipulate. However, it has also revealed that the use of optimizers without some care can give rise to problems, they being like any other specialized tools in this respect. This work continues, aiming to embed such optimization skills within an expert system as part of an overall package, enabling naval architects, unskilled in the use of optimizers, to have access to the technique. The specification of the relevant constraints and design variables for optimization is relatively straight-forward; the optimizers quickly reveal which constraints and variables are redundant. They also identify any missing constraints by their tendency to move into obviously absurd areas when fundamental limits are not applied. For example, if payload capacity is ignored and resistance selected as the objective, the optimizers quickly drive the displacement towards zero ! Indeed, the ability of optimization and expert systems techniques to afford a measure of tutorial assistance to new designers would appear to be one of the great advantages of such methods.

The choice of a suitable objective to measure quality in ship design, whether made by the user or an expert system, may be regarded as a severe limitation of any goal oriented design methods. However, such arguments apply equally well to all approaches to design. If a designer is unable to decide which of a number of competing designs is best, this probably demonstrates a lack of clarity in the original specification. Moreover, using numerical measures to evaluate competing designs forces the designer to distinguish between true objectives and simple constraints; is minimum cost the real aim or should the design merely attempt to get below some maximum cost threshold ? Conversely, is a given top speed essential or should this figure form part of a measure of merit, as something to be improved provided that the effects on other goals are not penal ? In fact, the overt requirement to quantify some measure of merit would, itself, seem to be a desirable pressure in moving naval architecture from the realms of art-form to science. Until the goals are specified in an essentially numerate fashion, the designer will be denying himself access to the best analytical tools on offer. It is hoped that the profession may come to embrace such modern *design* tools with the same alacrity as the now common place finite element *analysis* codes, which have been introduced in the last ten years. This may then go some way to redress the balance

RESULTS	Startship	Optimization with 2 variables				Opt. with 5 variables	
Method		APPROX	RANDOM *	SEEK	SEEK	RANDOM *	SEEK
Penalty Function		n/a	n/a	OPTIM1	OPTIM2	n/a	OPTIM1
R_T (KN)	1505.5	855.0	876.3	855.1	855.1	835.0	832.0
ⓜ	7.0	9.143	8.906	9.143	9.143	9.220	9.149
B_{WL}/T	4.0	3.410	3.399	3.410	3.411	3.303	3.396
C_P	0.57	0.57	0.57	0.57	0.57	0.558	0.550
$FLARE'_X$	0.0	0.0	0.0	0.0	0.0	4.7	0.4
L'_X	0.52	0.52	0.52	0.52	0.52	0.566	0.581
L_{WL} (m)	93.8	122.5	119.3	122.5	122.5	123.5	122.6
B_{WL} (m)	14.77	11.93	12.07	11.93	11.93	11.69	11.91
T (m)	3.693	3.499	3.551	3.499	3.499	3.541	3.505
D (m)	9.23	8.75	8.88	8.75	8.75	8.85	8.76
C_X	0.825	0.824	0.825	0.824	0.824	0.842	0.855
C_{WP}	0.722	0.722	0.722	0.722	0.722	0.720	0.719
B_{OA} (m)	14.77	11.93	12.07	11.93	11.93	13.15	12.02
BT_{WL}/B_{WL}	0.45	0.45	0.45	0.45	0.45	0.45	0.45
BT_{OA}/B_{OA}	0.95	0.95	0.95	0.95	0.95	0.95	0.95
$FLARE_X$ (°)	0.0	0.0	0.0	0.0	0.0	7.8	0.6
$FLARE_A$ (°)	35.0	30.8	30.8	30.8	30.8	30.0	30.7
ROF (°)	2.5	2.9	2.9	2.9	2.9	3.0	2.9
L_{MID} (m)	1.80	2.35	2.29	2.35	2.35	2.37	2.35
L_{KEEL} (m)	52.0	67.9	66.2	67.9	67.9	68.5	68.0
i_E (°)	13.0	8.1	8.4	8.1	8.1	7.9	8.1
$i_{E\ UD}$ (°)	25.0	16.0	16.7	16.0	16.0	15.7	16.0
$RAKE_F$ (°)	33.0	41.8	40.7	41.8	41.8	41.7	41.8
$RAKE_A$ (°)	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Initial GM (m)	2.01	0.70	0.71	0.70	0.70	0.70	0.70
Area 0-40° (m rad)	0.52	0.23	0.23	0.23	0.23	0.25	0.23
Area 30-40° (m rad)	0.23	0.11	0.11	0.11	0.11	0.13	0.12
Area 0-30° (m rad)	0.29	0.12	0.12	0.12	0.12	0.12	0.12
Angle Max GZ (°)	49	54	54	54	54	54	53
Max GZ (m)	1.57	0.92	0.92	0.92	0.92	0.98	0.97
L_{WL}/D	10.2	14.0	13.4	14.0	14.0	14.0	14.0
$\nabla_{ENC}(m^3)$	14065	13588	13605	13588	13588	14425	13647
Δ_D (t)	3274	3492	3462	3492	3492	3490	3491

Table 5 - Details of the Final Designs.
* modified shrinkage action, see text.

between the designer and the analyst who often criticizes his work.

REFERENCES

1. D. J. Andrews, "Creative Ship Design," *Trans. R.I.N.A.* 123 pp. 447-471 (1981).
2. D. J. Andrews, "An Integrated Approach to Ship Synthesis," *Trans. R.I.N.A.* 127 pp. 73-102 (1985).
3. A. J. Keane, "Ship Concept Design System," M.Sc. Thesis, Department of Mechanical Engineering, University College London (1981).
4. M. C. Eames and T. G. Drummond, "Concept Exploration - an Approach to Small Warship Design," *Trans. R.I.N.A.* 119 pp. 29-54 (1977).
5. I. M. Yuille, "The Forward Design System for Computer Aided Ship Design using a Mini-Computer," *Trans. R.I.N.A.* 120 pp. 323-341 (1978).
6. D. E. Calkins, "An Interactive Computer-Aided Synthesis Program for Recreational Powerboats," *Trans. S.N.A.M.E.*, pp. 49-87 (1983).
7. M. G. Parsons and K-P. Beir, "Microcomputer Software for Computer-Aided Ship Design," *Marine Technology* 24(3) pp. 246-264 S.N.A.M.E., (1987).
8. S.S. Tong, "Design of Aerodynamic Bodies Using Artificial Intelligence/Expert System Techniques," pp. 1-6 in *AIAA 23rd Aerospace Sciences Meeting*, , Reno (1985).
9. J. L. Alty and M. J. Coombs, *Expert Systems - Concepts and Examples*, NCC Publications (1984).
10. K. J. MacCallum and A. Duffy, "An Expert System for Preliminary Numerical Design Modelling," pp. 33-47 in *Proceedings of the Engineering Software Conference*, Computational Mechanics Centre (1985).
11. P. Mandel and R. Leopold, "Optimization Methods Applied to Ship Design," *Trans. S.N.A.M.E.*, pp. 477-521 (1966).
12. C. Gallin, "Which Way Computer-Aided Preliminary Ship Design and Optimization?," *Computer*

Applications in Shipping and Shipbuilding 2 pp. 393-403 North-Holland Publishing Company, (1974).

13. M.G. Parsons, "Optimization Methods For Use In Computer-Aided Ship Design," pp. 1-31 in *First Technology and Research (STAR) Symposium*, S.N.A.M.E., Washington D.C. (1975).
14. K.-P. Beier, H. Nowacki, C. Schubert, and A. Weichbrodt, "A General Purpose Software System for Optimization in Ship Design," pp. 111-116 in *Proceedings of the 2nd International Conference on Computer Applications in the Automation of Shipyard Operation and Ship Design*, North-Holland Publishing Company (1976).
15. M. S. Pantazopoulos, "An optimized CAD/CAE system for naval combatants," *The Naval Architect*, pp. E305-E308 (1985).
16. GINO, *GINO-F 2.7 User Manual*, CAD Centre Ltd., U.K. (1986).
17. Simpleplot, *Simpleplot Mark 2 Reference Manual*, Bradford University Software Services Ltd., U.K. (1987).
18. J. N. Siddall, *Optimal Engineering Design: Principals and Applications*, Marcel Dekker, Inc., New York (1982).
19. J. Holtrop and G. G. J. Mannen, "An Approximate Power Prediction Method," *International Shipbuilding Progress* 29(335) pp. 166-170 (1982).
20. J. Holtrop, "A Statistical Re-Analysis of Resistance and Propulsion Data," *International Shipbuilding Progress* 31(363) pp. 272-276 (1984).
21. A. J. Keane, "A Computer Based Method for Hull Form Concept Design: Applications to Stability Analyses," *Trans. R.I.N.A.* 130(A) pp. 61-75 (1988).
22. D. G. M. Watson and A. W. Gilfillan, "Some Ship Design Methods," *Trans. R.I.N.A.* 118 pp. 279-324 (1976).

APPENDIX - OPTIMIZATION METHODS (OPTIVAR)

The OPTIVAR package contains various optimization methods in a format enabling simple use of a variety of modern techniques. These methods may be classified as: a) unconstrained nonlinear methods; b) constrained nonlinear methods combined with penalty functions; c) single-variable minimization; d) linear-programming. Of these, the constrained and unconstrained nonlinear methods are relevant to the problems being investigated here. The constrained nonlinear methods available are:-

(1) Minimization by the method of successive linear approximation (APPROX).
(2) Minimization by random exploration with shrinkage (RANDOM).

The unconstrained nonlinear methods are a group of different strategies all using the same interchangeable penalty functions to deal with constraint violations:-

(1) Minimization by adaptive random search (ADRANS).
(2) Minimization by Davidson-Fletcher-Powell method (DAVID).
(3) Minimization by Fletcher's 1972 method (FLETCH).
(4) Minimization by Jacobson and Oksman method (JO).
(5) Minimization by Powell's direct search method (PDS).
(6) Minimization by Hooke and Jeeves direct search (SEEK).
(7) Minimization by the simplex method (SIMPLX).

The penalty functions available are:-

(1) One pass external function (OPTIM1).
(2) Fiacco-McCormick combined external and internal function (OPTIM2).
(3) Powell's function (OPTIM3).
(4) Schuldt's function (OPTIM5).

All of these methods are briefly described by Siddall[18], where further references to the original works may also be found. For the present example, the two constrained methods were adopted together with one of the unconstrained methods (SEEK) combined with the first two penalty functions (OPTIM1 and OPTIM2). These represent four different strategies: APPROX is the quickest method if the function is smooth enough to be approximated to a linear one in the region of investigation. It tests the vertices of the n-dimensional form created from the objective function allowing for planes formed by linearizing the various constraints. RANDOM is a random 'shotgun' search method, it is very slow, but also potentially the most reliable. If used in its purest form it will always find the optimum and so is used as a reference for the evaluation of the other strategies (a shrinkage mechanism is invoked by default with this method to narrow the area for searching but this introduces the risk of finding a false optimum). Finally, one of the unconstrained nonlinear methods was selected. The Hooke and Jeeves method was chosen since it is both well known and typical of an heuristic approach. It is a direct search strategy that tests variables systematically and then decides the directions and sizes of steps to be taken using various comparisons. The penalty functions employed are basically of two kinds: a one pass external function, which penalizes the function only when the constraints are violated, and then extremely strongly; or multiple pass functions which penalize the function both inside the feasible region as the constraints are being approached as well as in the unfeasible area. The Fiaccao-McCormick, Powell's and Schuldt's functions are multiple pass functions having slightly different formulations, but all rely on penalties that become increasingly severe as repeated optimizations are carried out. These later types of penalty function are useful when the function being optimized cannot cope with variables that lead to unfeasible results, making the system 'crash'. However, since they are mild in action and try to avoid going near unfeasible regions of the function where optima often lie, this approach tends to be slower than the first. For robust systems, i.e., systems that can cope with calculations in unfeasible areas, the one pass external function proves to be the quickest.

All the methods try to solve the problem which may be described in mathematical terms as follows:-

find

$$U = U(x_1, x_2, \cdots, x_n) = \text{minimum}$$

subject to

$$\psi_i(x_1, x_2, \cdots, x_n) = 0 \qquad i = 1, m$$

and

$$\phi_j(x_1, x_2, \cdots, x_n) \geq 0 \qquad j = 1, p$$

where U is the objective function, $x_1, \cdots, x_n$ are the n variables that may be changed by the optimizer (the trial vector), $\psi_i (i=1,m)$ are m equality constraints and $\phi_j (j=1,p)$ are p inequality constraints. (n.b. random searches cannot deal with equality constraints, but these are not applicable here.)

A.1. Method of Successive Linear Approximation (APPROX)

The method employed is a successive linear approximation developed by Griffith and Stewart. Starting at an initial feasible point $(x_1^0, x_2^0, \cdots, x_n^0)$ the functions are approximated by expansion in a Taylor's series about x_i^0 in which terms above linear are dropped. As the problem has been converted to linear form, it can be solved by linear programming. The linearization is applied over a finite range that is subsequently moved and reduced as the optimization proceeds. The constraints, which are also linearized, are used to further reduce the linearized region. This method is very fast but has final convergence problems. These problems are exacerbated by singularities or excessive constraints.

A.2. Random Exploration with Shrinkage (RANDOM)

This method consists of a random search for the minimum, using a shotgun technique, with iterative shrinkage. Random points for each variable x_1 to x_n are generated from the expression $x_i = l_i + r_i(u_i - l_i)$ where l_i is the estimated lower limit for x_i, u_i is the estimated upper limit to x_i and r_i is a random number uniformly distributed between zero and one. Any generated point that violates an inequality constraint is discarded before the objective function is evaluated. If the constraints are violated a certain number of times consecutively, by default 300, the process stops.

The search is begun by evaluating a number of random points using the above equation, this number being a multiple of the number of variables. From these, the best results are selected and used as the basis for a new and shrunken range for each variable. The number of best results is defined by dividing the number of random points by a shrinkage factor. Within this new space, new random points are evaluated. These, plus the previous best results, are sorted to yield a new set of best results and a new shrunken space. The process is repeated until the range of each variable is acceptably small, or until the range has been shrunken a certain number of times, 1000 being the default. If any set of random points does not include one from near the optimal region the shrinkage mechanism can reject the area around the optimum and thus carries the risk of finding a false optimum. However, this mechanism does significantly improve the speed of the search, particularly over high dimensional spaces. When the problem is highly constrained shrinkage must be treated with caution.

A.3. Hooke and Jeeves Direct Search (SEEK)

This method uses the following heuristic algorithm for minimization:-

(1) An arbitrary starting point must be selected, called the initial base point, at which the objective function U is evaluated.

(2) An exploratory search is begun by increasing x_1 by a predetermined step length. If this improves the value of U, it is retained, if not, a corresponding reduction is made. If both fail, no change is made. A similar exploration is made for x_2 and so on. The final result of such an exploratory search is called a base point.

(3) A pattern move is made by changing each variable from the last base point an amount equal to the difference between the new and previous base point. This difference will commonly include a previous pattern move.

(4) If the pattern move fails to improve U, it is cancelled and replaced by a new search. If it succeeds, it is followed by a new search.

(5) The iteration continues until the exploratory search fails to locate a better point. The step length is then reduced an arbitrary amount, and the search is repeated. After each failure the step length is reduced an additional fraction until it reaches some predetermined minimum, when it is assumed that the optimum has been reached.

A.4. Penalty Functions

A penalty function is a simple technique used to distort the optimization function in order to force the search towards feasibility, when a constraint is, or is about to be, violated. The simplest form of penalty function is:

$$U_p = U(x_1, x_2, \cdots, x_n) + r\sum_{i=1}^{m} |\psi_i| + r\sum_{j=1}^{p} |\langle\phi_j\rangle|$$

where U_p is the penalized objective function which is minimized instead of U, r is a large number and the symbol $\langle\alpha\rangle$ has the meaning

$$\langle\alpha\rangle = \begin{cases} \alpha & \text{if } \alpha \le 0 \\ 0 & \text{if } \alpha > 0 \end{cases}$$

This penalty function is used as the one pass external function with $r = 10^{20}$. This is very severe and sometimes the search stalls, particularly if it must follow along the constraint line to reach the optimum. Despite this disadvantage, the strategy is often quite successful, and requires a minimum of computer time when it works. (When using this function, the SEEK algorithm uses 100 random shotgun steps when it fails to find a better point, to overcome stalling.)

The other penalty functions in the suite soften this severe distortion and to try to achieve less risk of premature stalling of the search. The one pass penalty function is only active when constraints are violated and the search is in an infeasible region. Functions having this characteristic are called external penalty functions. Conversely, internal functions are active only in the feasible region, so that the surface is distorted when the inequality constraint line is approached, and the search is 'warned' that there is trouble ahead. The Fiacco-McCormick function combines these approaches and takes the form :-

$$U_p = U(x_1, x_2, \cdots, x_n) + \frac{1}{r}\sum_{i=1}^{m}\psi_i^2 + r^2\sum_{j=1}^{p}\frac{1}{\phi_j^S} + \frac{1}{r}\sum_{j=1}^{p}\langle\phi_j\rangle^2$$

Here the symbol $\langle\ \rangle$ has the previous meaning indicating an unsatisfied constraint, while superscript S indicates a satisfied one. The process begins with $r=1$ and a sequence of optimization problems are solved in which r is progressively reduced. To justify this expression, consider the effect on a two-dimensional problem with one inequality constraint. If the interior term is considered, (r^2/ϕ_1^S), with $r=1$, it will push the surface upwards asymptotically to the constraint line, beginning the distortion some distance into the feasible region. As the constraint line is approached ϕ_1^S becomes smaller and smaller. It is apparent that a false constrained optimum point will be established well away from the true location at the constraint line. As r becomes smaller, the effect of $1/\phi_1^S$ is increasingly reduced, the surface is sharpened, and the false constrained optimum point will approach the true one.

The external term $\langle\phi_1\rangle^2/r$ has a similar effect in the exterior region. With $r=1$, there is a gentle distortion of the surface remote from the constraint line which becomes small as ϕ_1

approaches zero at the constraint line. A false unconstrained optimum point is created near to the feasible region. Subsequent reductions in r increase the effect of the penalty term, sharpening the curve, and shifting the false, unconstrained optimum toward the true constrained location.

A.5. Remarks

These four approaches have various advantage and disadvantages. RANDOM, as has already been mentioned, may be very slow particularly if the feasible region is rather restricted and there will then be a high risk of a large number of infeasible trials. On the other hand, it does not hang up on false or local optima. For this reason, it is a good method for checking the results of other approaches. Care must be taken to ensure that the shrinkage mechanism does not reject the true optimum. With APPROX, if the function and boundaries formed by the constraints are not amenable to linear approximation, although finding the optimal region quickly, it will have great difficulty in converging. Testing of APPROX has shown that even functions such as the one being studied here can sometimes be successfully approximated as linear over the small ranges required. However, highly constrained functions or those containing singularities are not handled well. With SEEK, the Hooke and Jeeves method, the main difficulty arises in highly constrained problems. The search can get stuck on constraints because of the fixed orientation of its search coordinates. If the contour lines and the inequality constraint lines happen to have certain orientations, and the search approaches from particular directions, the search may become stalled. A new starting point may well alleviate the problem. Some types of 'soft' penalty functions may also be helpful in extending the search. A local random search may help the search to jump out of trouble and this is included by default with the one pass penalty function.

REFERENCES

1 Rawson,K.J. and Tupper,E. "Basic Ship Theory", Longmans (1968)

2 Calkins, D.E. "Ship Design Synthesis Model Morphology", 3rd International Ship Design Conference, S.N.A.M.E. Pittsburgh, June 1988

3 Andrews,D.J. "Creative Ship Design", Trans. R.I.N.A. 123 pp. 447-471 (1981)

4 Andrews,D.J. "An Integrated Approach to Ship Synthesis", Trans.R.I.N.A. 127 pp. 73-102 (1985)

5 Comstock, J.P. "Principles of Naval Architecture", S.N.A.M.E. (1967)

6 Lamb, T. "A Ship Design Procedure", Marine Technology 6(4), Oct. 1969

7 Watson, D.G.M. and Guilfillan A.W. "Some Ship Design Methods", Trans. R.I.N.A. 118 pp. 279-324 (1976)

8 Oossanen, P.Van "Resistance Prediction of Small High-Speed Displacement Vessels: State of the Art", Symposium on the 'Impact of 200 Mile Economic Zones' R.I.N.A. Australia, Nov. 1979

9 Holtrop, J. "A Statistical Analysis of Performance Test Results", International Shipbuilding Progress 24(270) pp. 23-28, Feb. 1977

10 Holtrop, J. "Statistical Data for Extrapolation of Model Performance Tests", International Shipbuilding Progress 25(285) pp. 122-126, May 1978

11 Holtrop, J. and Mannen, G.G.J. "A Statistical Power Prediction Method", International Shipbuilding Progress 25(290) pp. 253-256, Oct. 1978

12 Holtrop, J. "An Approximate Power Prediction Method", International Shipbuilding Progress 29(335) pp.166-170 (1982)

13 Holtrop, J. "A Statistical Re-Analysis of Resistance and Propulsion Data", International Shipbuilding Progress 31 (363) pp. 272-276 (1984)

14 Oortmerssen, G. Van "A Power Prediction Method and its Applications to Small Ships", International Shipbuilding Progress 18(207) pp.397-415 (1971)

15 Sarchin, T.H. and Goldberg, L.L. "Stability and Buoyancy Criteria for US Naval Surface Ships", Trans. S.N.A.M.E. pp.418-458 (1962)

16 Farrar, A. "Computer aided design for ships", Ship & Boat International pp.39, Sept. 1989

17 Snaith, G.R. and Parker, M.N. "Ship Design with Computer Aids" Trans. NECIES 88 pp.151-172 (1972)

18 Eames, M.C. and Drummond, T.G. "Concept Exploration - an Approach to Small Warship Design", Trans. R.I.N.A. 119 pp. 29-54 (1978)

19 Yuille, I.M. "The Forward Design System for Computer Aided Ship Design using a mini-computer", Trans. R.I.N.A. 120 pp. 323-341 (1978)

20 Parsons, M.G. and Beier, K.-P. "Microcomputer Software for Computer-Aided Ship Design", Marine Technology 24(3) pp.246-264 S.N.A.M.E. (1987)

21 Calkins, D.E. "An Interactive Computer-Aided Synthesis Program for Recreational Powerboats", Trans. S.N.A.M.E. pp. 49-87 (1983)

22 Keane, A.J. "A Computer Based Method for Hull Form Concept Design: Applications to Stability Analyses", Trans. R.I.N.A. 130(A) pp. 61-75 (1988)

23 Siddall, J.N. "Optimal Engineering Design: Principles and Applications", Marcel Dekker, Inc., New York (1982)

24 Gill, P.E. and Murray, W. "Numerical Methods for Constrained Optimisation", Academic Press (1974)

25 Murray, W. "Methods for Constrained Optimisation", Dixon, L.C.W. 'Optimisation in Action', Chapter 12, Academic Press (1976)

26 Goldberg, D.E. "A Tale of Two Problems: Broad and Efficient Optimization using Genetic Algorithms", Department of Engineering Mechanics, The University of Alabama

27 Poole, I. and Adams, H. "Lapwing - A Trainable Image Recognition System for the Linear Array Processor", 4th International Conference of Pattern Recognition, March 1988

28 Poole, I. "GAGA - A Genetic Algorithm for General Applications", Internal Note No. 2251, Department of Computer Science, University College London, Apr. 1988

29 Kirkpatrick, S., Gelatt Jr, C.D. and Vecchi, M.P. "Optimization by Simulated Annealing", Science 220(4598) pp.671-680, May 1983

30 Parsons, M.G. "Optimization Methods for use in Computer-Aided Ship Design", pp. 1-31 in First Technology and Research (STAR) Symposium, S.N.A.M.E., Washington D.C. (1975)

31 Bales, N.K. "Optimizing the Seakeeping Performance of Destroyer-Type Hulls", DTNSRDC, Prepared for the 13th Symposium on Naval Hydrodynamics, Tokio, Oct. 1980

32 Mandel, P. and Leopold, R. "Optimization Methods Applied to Ship Design", Trans. S.N.A.M.E., pp. 477-521 (1966)

33 Beier, K.-P., Nowacki, H., Schubert, C. and Weichbrodt, A. "A General Purpose Software System for Optimization in Ship Design", pp. 111-116 in Proceedings of the 2nd International Conference on Computer Applications in the Automation of Shipyard Operation and Ship Design, North-Holland Publishing Co (1976)

34 Pantazopoulos, M.S. "An optimized CAD/CAE system for naval combatants", The Naval Architect, pp. E305-E308, Jul/Aug. 1985

35 Lyon, T.D. and Mistree, F. "A Computer-Based Method for Preliminary Design of Ships", Journal of Ship Research 29(4) pp. 251-269 S.N.A.M.E., Dec. 1985

36 Buchanan, B.G. and Shortliffe, E.H. "Rule-Based Systems", Addison-Wesley Publishing Co (1985)

37 Waterman, D.A. "A Guide to Expert Systems", Addison-Wesley Publishing Co (1986)

38 Johnson, L. and Keravnou,E.T. "Expert Systems Architectures" New Generation Computing Series, Kogan Page (1988)

39 Hayes-Roth, F., Waterman, D.A. and Lenat, D.B. "Building Expert Systems", Addison-Wesley Publishing Co (1983)

40 Jackson, P. "Introduction to Expert Systems", Addison-Wesley Publishing Co (1986)

41 Alty, J.L. and Coombs, M.J. "Expert Systems - Concepts and Examples", NCC Publications (1984)

42 Harman, P., Maus R. and Morrisey, W. "Expert Systems Tools and Applications", John Wiley and Sons, Inc., New York (1988)

43 Kowalik, J.S. "Coupling Symbolic and Numeric Computing in Expert Systems", Elsevier Science Publishers B.V. (1986)

44 Tong, S.S. "Design of Aerodynamic Bodies Using Artificial Intelligence/Expert System Techniques", pp. 1-6 in AAIA 23rd Aerospace Sciences Meeting, Reno, Nevada, Jan. 1985

45 MacCallum, K.J. "Creative Ship Design by Computer", pp.55-62 in Computer Applications in the Automation of Shipyard Operations & Ship Design IV, North-Holland (1982)

46 MacCallum, K.J. and Duffy, A. "An Expert System for Preliminary Numerical Design Modelling", pp. 33-47 in Proceedings of the Engineering Software Conference, Computational Mechanics Centre (1985)

47 Welsh, M., Buxton,I.L., Hills, W. "The Application of an Expert System to Ship Concept Design Investigations", Paper No. 10, R.I.N.A. Spring Meetings, London (1990)

48 Hills, W. and Buxton, I.L. "Integrating Ship Design and Production Considerations During the Pre-Contract Phase", The Naval Architect pp. 189-209, Oct. 1989

49 Betts, C.V. "Engineering Design at Sea ?", Inaugural Lecture at University College London, 30th October 1986

50 Keane, A.J., Price, W.G. and Schachter, R.D. "Optimization Techniques in Ship Concept Design", Paper No. 11, R.I.N.A. Spring Meetings, London (1990)

51 GINO, GINO-F 2.7 User Manual, CAD Centre Ltd, U.K. (1986)

52 Simpleplot, Simpleplot Mark 2 Reference Manual, Bradford University Software Services Ltd, U.K. (1987)

53 Keane, A.J. "GENDAT - The General Purpose Data Handler User Guide", Brunel University internal publication (1988)

54 Expert Systems 87 Conference - Notes from Tutorial I, Brighton, Dec. 1987

55 Leonardo Version 2, Reference Manual, User Guide and Technical Summary, Creative Logic, Brunel Science Park, U.K. (1989)

56 Mischke, C.A. "An Introduction to Computer Aided Design", Prentice Hall (1968)

57 Haefele, J.W. "Creativity and Innovation", Chapman & Hall, London (1982)

58 ITTC Dictionary of Ship Hydrodynamics, Marine Technology Monograph No.6, R.I.N.A. (1978)

59 International Towing Tank Conference Standard Symbols 1976, B.S.R.A. Technical Memorandum, T.M. No. 500, The British Ship Research Association, May 1976

60 Meissner L.P. and Organick E.I. "Fortran 77: featuring structured programming", Addison-Wesley Publishing Co, 1984

61 Waite, M., Martin, D. and Prata, S. "Unix Primer Plus", Howard W. Sams & Co (1986)

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisors, Professor W.G. Price for the benefit of his knowledge and experience and Dr. A.J. Keane for his guidance, encouragement and support, throughout my work.

I am grateful to the National Council of Research (CNPq - Brazil) for their financial support and for the attention and help given by Professor E. Oswaldo-Cruz, their representative in London, during the period of this research.

I would like to thank Dr. P. Temarel for his encouraging advice and enthusiasm. I gratefully acknowledge Professor L. Johnson (Dept. of Computer Science, Brunel University) and Mr. S. Pfeifer for their valuable help in Expert Systems.

I am indebted to the staff of the Department of Mechanical Engineering, who have provided a friendly and supportive environment in which to work, especially Mrs. O.I. Rolph for her expert typing and Mr. R.T. Murdock for his constant support, and my colleagues of the Marine Group: Mr. S.M. Cannon, Dr. Y. Wu, Mr. S. Aksu, Mr. Y.S. Kwon, Mr. M. Tan, Mr. X.J. Wu, Mr. A. Ergin and Mr. S. Miao, I am grateful to them all for their interest, co-operation, advice and practical help.

I would like to give my special thanks to Professor D.E. Calkins, of the University of Washington, to whom I owe the suggestion of my specialisation and valuable references, Professor P.D. Martins (COPPE/UFRJ), for the invaluable discussions, and Professors C.L. Barauna Vieira, M.A.S. Neves, F.C. Martins Jr. and C.L.A. da Conceicao, all from COPPE, Federal University of Rio de Janeiro, for being instrumental in encouraging me to undertake this research at Brunel University.

Finally, I would like to thank Maria Luiza, Daniel and Roberta, my wife and children, without whose patience, understanding and encouragement this work would have never been completed.