NOTE TO USERS

This reproduction is the best copy available.

UMI[®]

**MULTIDISCIPLINARY OPTIMIZATION
OF
CONCEPTUAL AIRCRAFT DESIGN**

By

Cheryl A. Eisler, B. Eng.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Applied Science

**Department of Mechanical and Aerospace Engineering
Ottawa-Carleton Institute of
Mechanical and Aerospace Engineering**

**Carleton University
Ottawa, Ontario
November 2003**

Copyright © 2003

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisistons et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-89834-2

Our file Notre référence

ISBN: 0-612-89834-2

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

ABSTRACT

Multidisciplinary design optimization (MDO) is a methodology for the design of complex, interdependent systems that is growing in popularity for use in the early phases of aircraft design. The conceptual and preliminary design phases are particularly well suited to the application of MDO considering that the historical approach to aircraft design is broken into specific disciplinary areas, such as aerodynamics, structures, and propulsion. The advantages of this methodology include decreasing the time spent in early design phases, increasing the number of concepts evaluated, increasing the level of detail to which each concept is examined, and decreasing project costs through the elimination of suboptimal designs early in the process. By automating the analysis and the optimization, off-the-shelf optimization and design software can be integrated together. The time to produce a feasible and locally optimal design is reduced and the information is stored in a format compatible with more detailed design software.

This thesis examines the history of automated MDO as applicable to conceptual aircraft design, and outlines the development of a new software tool that addresses some of the issues of flexibility and visualization that have arisen in the past. Two test cases were used to evaluate the performance of both the analysis methods and optimization algorithms, as compared to results of the traditional design methodology. Consequently, recommendations are made to help improve the system model and the objectives of the optimization.

ACKNOWLEDGEMENTS

I would like to thank the following people for their generous contribution and support towards my thesis work:

- Ann Eisler, my mother, for always encouraging me to establish my own path and not limit myself in any way.
- Alex Samousevitch, my significant other, for his willingness to listen and asking the “hard” questions.
- Brendan Simons, my friend, for his readiness to double-check my computations and showing me new ways of looking at problems.
- Paul Straznicky, my thesis supervisor, for his support and understanding through good times and bad.
- Engineous Software, Inc., provider of iSIGHT, for their interest in and commitment to establishing a new area of research at Carleton University.

Without you all, I would not have been able to come this far.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENTS	ii
LIST OF FIGURES	x
LIST OF TABLES	xiii
NOMENCLATURE	xiv
GLOSSARY OF TERMS	xviii
1. Introduction	1
1.1 Multidisciplinary Design Optimization	1
1.2 Design Development	2
1.3 From Conceptual Design Iteration to Automation	5
1.4 The Nature of the Problem	7
1.5 Issues with Early Automated MDO	9
1.6 Multidisciplinary Optimization of Conceptual Aircraft Design	10
2. Problem Formulation	13
2.1 The Objective Function	13
2.1.1 Penalty Functions	14
2.1.2 Physical Programming	15
2.2 The Constraints	15
2.3 The Initial Design Point	16
2.4 The Optimal Solution	17

2.4.1 Local Optima	17
2.4.2 Pareto Optimal Solutions	18
3. MDO Components	19
3.1 Mathematical Modelling	19
3.2 Approximation Techniques	20
3.2.1 Generic Approximations	21
3.2.2 Corrected Approximations	22
3.2.3 Reduced Order Approximations	23
3.2.4 Quasi-Approximation Techniques	23
3.3 Optimization Algorithms	24
3.3.1 Enumeration	24
3.3.2 Stochastic Search Methods	25
3.3.3 Optimality Criteria	27
3.3.4 Gradient-Based Search Methods	28
3.3.5 Pattern Search Methods	30
3.4 System Sensitivity Analysis	31
3.4.1 Sensitivity Analysis Techniques	32
3.4.2 Global Sensitivity Equations	34
3.5 Decomposition Methods	37
3.5.1 All-In-One	38
3.5.2 Hierarchic Decomposition	39
3.5.3 Non-Hierarchic Decomposition	40
3.5.4 Automated Decomposition	43

3.6 Dealing With Uncertainty	44
3.7 Human Interface.....	45
3.7.1 Data Management	45
3.7.2 Visualization	46
3.7.3 Qualitative Aspects	48
3.7.4 Future Functionality	48
4. Specific Issues With Respect to Conceptual Aircraft Design.....	49
4.1 Design Variables, Constraints and Measures of Merit	49
4.2 Analysis Fidelity	52
4.3 Analysis Modules.....	53
4.4 Geometry.....	54
4.5 Scaling Laws.....	55
4.6 Cost Estimation.....	55
4.7 Manufacturing and Assembly Issues	57
5. Aircraft Synthesis Software	59
5.1 AAA (University of Kansas)	59
5.2 G.A.-CAD (Design, Analysis, and Research Corp.).....	60
5.3 RDS (Conceptual Research Corporation).....	60
5.4 ASCENT (Lockheed Martin).....	61
5.4.1 ASSET (Lockheed Martin Skunk Works)	61
5.4.2 ACSYNT (ACSYNT Institute).....	61
5.5 PASS (Stanford University).....	63

5.6 CASDAT (Georgia Institute of Technology and Air Force Research Laboratory)	65
6. Conceptual Aircraft Design Optimization Software	66
6.1 OPDOT (NASA Langley)	66
6.2 FLOPS (Kentron International, Inc.)	67
6.3 GASP + OPT (Lockheed-Georgia)	68
6.4 CASTOR (Loughborough University)	69
6.5 ADAS (Delft University)	70
6.6 TASOP (BAe Commercial Aircraft Ltd.)	71
6.7 CAPDA (Delft University)	73
6.8 PASS + NPSOL + GA (Stanford University)	74
6.9 ACSYNT (ACSYNT Institute)	74
6.10 RDS-Professional (Conceptual Research Corporation)	75
6.11 AIDA (Delft University)	76
6.12 BLISS (NASA Langley)	77
6.13 RCD (Lockheed Martin Skunk Works)	78
7. Project Requirements	80
7.1 General MDO Software Requirements	80
7.2 Software Requirements for Conceptual Aircraft Design	82
8. The Multidisciplinary Optimization of Conceptual Aircraft (MOCA) Project	83
8.1 MOCA Software Structure	83
8.2 Current Project Development	89
8.2.1 Software Structure Development	89

8.2.2 Module Development.....	92
8.2.3 Model Development.....	92
9. Project Direction	93
10. Software Selection	94
10.1 Selection of an Optimization Framework	94
10.1.1 Software Requirements for an Optimization Framework.....	95
10.1.2 Hardware Requirements for an Optimization Framework.....	99
10.1.3 Optimization Framework Comparison	100
10.1.4 Feature-Based Trade-off Study.....	101
10.1.5 Qualitative Trade-off Study	101
10.1.6 Composite Trade-off Values.....	101
10.1.7 Conclusions.....	104
10.1.8 Results.....	105
10.2 Selection of an Analysis Environment.....	106
10.2.1 Requirements for an Analysis Environment.....	106
10.2.2 Analysis Environment Selection.....	107
10.3 Selection of a CAD Program	108
10.3.1 Requirements for a CAD Program.....	108
10.3.2 CAD Program Selection	109
11. The MOCA Software Tool	110
11.1 Optimization of an Existing Aircraft Design	110
11.1.1 Propulsion	120
11.1.2 Geometric Configuration	121

11.1.3 Aerodynamics	124
11.1.4 Initial Sizing.....	124
11.1.5 Performance Matching.....	124
11.2 Optimization Problem Formulation	128
11.3 Optimization Algorithms and Approximation Techniques.....	128
11.4 Optimal Design Visualization.....	129
11.5 MOCA Program Feature Summary	130
12. Testing of the MOCA Software Tool	131
12.1 Test Case 1	131
12.1.1 Sample Mission Requirements	132
12.1.2 Geometric Configuration	133
12.1.3 Trial 1.....	135
12.1.4 GeoSurv UAV Drag Estimation	137
12.1.5 Trial 2.....	139
12.2 Test Case 2.....	149
12.2.1 Sample Mission Requirements	149
12.2.2 Geometric Configuration	150
12.2.3 Trial 1	152
13. Conclusions.....	160
13.1 Current Software Development	160
13.2 Selected Software Performance	163
13.3 Software Tool Performance	164
13.4 Further Software Development.....	165

14. Future Project Development	166
REFERENCES	169
Appendix A: Optimization Framework Cost Information (US \$).....	182
Appendix B: Optimization Framework Vendor Contact Information.....	185
Appendix C: Integrating Matlab as a Simcode into iSIGHT	187
Appendix D: Integrating Pro/Engineer as a Simcode into iSIGHT	204
Appendix E: Engine and Boom Placement.....	210

LIST OF FIGURES

Figure 1- 1: Design Development Phases.....	2
Figure 1- 2: Design Knowledge and Freedom.....	3
Figure 1- 3: Effect of MDO on Design Phases.....	4
Figure 1- 4: Design Space Shapes	8
Figure 2- 1: Global and Local Optima.....	18
Figure 3- 1: Approximation Processes.....	21
Figure 3- 2: An Example of Basis Reduction.....	23
Figure 3- 2: Hierarchic System Decomposition.....	39
Figure 3- 3: Non-Hierarchic System Decomposition	40
Figure 3- 4: BLISS Decomposition Cycle.....	43
Figure 3- 5: Visualization Techniques.....	47
Figure 4- 1: “Disciplines”	52
Figure 4- 2: Life Cycle Cost Breakdown.....	56
Figure 5- 1: PASS System Structure.....	63
Figure 5- 2: CASDAT System Structure	65
Figure 6- 1: OPDOT System Structure.....	67
Figure 6- 2: FLOPS System Structure	68
Figure 6- 3: GASP + OPT System Structure	69
Figure 6- 4: ADAS System Structure	70
Figure 6- 5: TASOP System Structure.....	72
Figure 6- 6: CAPDA System Structure.....	73

Figure 6- 7: RDS-Professional Optimization System Structure	75
Figure 6- 8: AIDA System Structure	76
Figure 6- 9: BLISS System Structure	78
Figure 6- 10: RCD System Structure	79
Figure 8- 1: MOCA Project Logo	83
Figure 8- 2: MOCA System Flow Diagram.....	85
Figure 8- 3: Aircraft Type Selection and its Relation to the Module Categories	86
Figure 8- 4: Modified System Structure	91
Figure 10- 1: Optimization Framework Spectrum.....	100
Figure 11- 1: The MOCA System Structure	111
Figure 11- 1: The Initial MOCA GUI.....	112
Figure 11- 2: The Mission Profile GUI, Including the Help Dialogue	113
Figure 11- 3: The Geometric Configuration GUI	114
Figure 11- 4: Sample Aircraft Configurations	116
Figure 11- 5: Lifting Surface Planform	119
Figure 11- 6: Performance Matching Constraint Violation, No Flaps.....	126
Figure 11- 7: Performance Matching Constraint Violation, Slats and Triple Slotted Flaps	127
Figure 12- 1: Effect of Equivalent Skin Friction and Wetted Area on Equivalent Parasite Area for Single Engine Propeller Driven Airplanes	139
Figure 12- 2: Comparison of 3-D Conceptual Designs of the GeoSurv UAV	141
Figure 12- 3: Planform Comparison of the Initial GeoSurv UAV Models.....	142
Figure 12- 4: Planform Change As a Result of Wing Span Variation.....	144

Figure 12- 5: Planform Change As a Result of Wing Span and Aspect Ratio Variation	146
Figure 12- 6: Performance Matching for the Optimal Design Point with Triple Slotted Flaps plus Slats for the Sea Level Stall Condition.....	148
Figure 12- 6: Performance Matching for the Optimal Design Point at $C_f = 0.0030$ and No Flaps for the Sea Level Stall Condition	154
Figure 12- 7: Performance Matching for the Optimal Design Point at $C_f = 0.0030$ and Triple Slotted Flaps Plus Slats for the Sea Level Stall Condition	154
Figure 12- 8: Performance Matching for the I-GNAT UAV Design Point	155
Figure 12- 9: The Initial Solid Model of the I-GNAT UAV	156
Figure 12- 10: The I-GNAT UAV	156
Figure 12- 12: Engine Fit in Initial Fuselage	157
Figure 12- 13: Engine Fit in Modified Fuselage.....	158
Figure 12- 11: The Modified Fuselage Solid Model of the I-GNAT UAV	158

LIST OF TABLES

Table 10- 1: Feature-Based Trade-off Study	102
Table 10- 2: Qualitative Trade-off Study.....	103
Table 10- 3: Composite Trade-off Results.....	103
Table 11- 1: Geometric Configuration Options	115
Table 11- 2: Geometric Configuration Parameters	118
Table 12- 1: GeoSurv UAV Geometric Configuration Options	133
Table 12- 2: GeoSurv UAV Geometric Configuration Parameter Values	134
Table 12- 3: Initial Sizing Comparison.....	136
Table 12- 4: Parasite Drag Coefficient Drag Breakdown	137
Table 12- 5: Gross Takeoff Weight Variation with Optimization Algorithm and T/W	140
Table 12- 6: Gross Takeoff Weight Variation with Optimization Algorithm, T/W, and Wing Span.....	144
Table 12- 7: Gross Takeoff Weight Variation with Optimization Algorithm, T/W, Wing Span, and Aspect Ratio	145
Table 12- 8: I-GNAT Geometric Configuration Options	150
Table 12- 9: I-GNAT Geometric Configuration Parameter Values.....	151
Table 12- 10: Gross Takeoff Weight Variation with C_f and T/W	152
Table 12- 11: Modified Fuselage Parameters	157

NOMENCLATURE

AAA	Advanced Aircraft Analysis
AAO	All-At-Once (Optimization)
ACSYNT	Aircraft Synthesis Tool
ADAS	Aircraft Design and Analysis System
AIDA	Artificial Intelligence-supported Design of Aircraft
ASCII	American Standard Code for Information Interchange
ASSET	Advanced System Synthesis and Evaluation Technique
BLISS	Bi-Level Integrated System Synthesis
CAD	Computer-Aided Drafting
CAM	Computer-Aided Manufacturing
CAPDA	Computer-Aided Procedure for the Conceptual Design of Aircraft
CASDAT	Conceptual Aerospace Systems Design and Analysis Toolkit
C_D	Drag Coefficient
C_{D_0}	Parasite Drag Coefficient
C_f	Skin Friction Coefficient
C_L	Lift Coefficient
CE	Concurrent Engineering
CO	Collaborative Optimization
CSSO	Concurrent Subspace Optimization
CSSUA	Concurrent Subsystem Uncertainty Analysis
DeMAID	Design Manager's Aid for Intelligent Decomposition

DOC	Direct Operating Cost
DOE	Design of Experiments
DSM	Design Structure Matrix
f	Equivalent Parasite Area
FARE	Income Required per Flight for a Fixed Return on Investment
FIO	Fully Integrated Optimization
FLOPS	Flight Optimization System
ft	Unit of Measure of Distance, Feet
GA	Genetic Algorithm
G.A.-CAD	General Aviation Computer Aided Design
GASP	Generalized Aircraft Sizing Program
GeoSurv UAV	Geophysical Survey Unmanned Aerial Vehicle
GHz	Gigahertz
GRG	Generalized Reduced Gradient Method
GSE	Global Sensitivity Equations
GUI	Graphical User Interface
hp	Unit of Measure of Power, Horsepower
hr	Unit of Measure of Time, Hour
IBL	Import Blend File
IDF	Individual Discipline Feasible
IGES	Initial Graphics Exchange Standard
IPPD	Integrated Product and Process Development
ISA	International Standard Atmosphere

KKT	Karush-Kuhn-Tucker Conditions
kts	Unit of Measure of Speed, Knots
lbs	Unit of Measure of Force, Pounds
L/D	Lift to Drag Ratio
MB	Megabytes
MCS	Monte Carlo Simulation
MDA	Multidisciplinary Design Analysis
MDF	Multidisciplinary Feasible
MDO	Multidisciplinary Design Optimization
min	Unit of Measure of Time, Minute
MMFD	Modified Method of Feasible Directions
MOCA	Multidisciplinary Optimization of Conceptual Aircraft
MOM	Measure of Merit
n-D	n-dimensional
nm	Unit of Measure of Distance, Nautical Miles
OML	Outer Mould Line
OPDOT	Optimum Preliminary Design of Transports
PC	Personal Computer
RAE	Royal Aircraft Establishment
RAM	Random Access Memory
RCD	Rapid Conceptual Design
ROI	Return on Investment
SAND	Simultaneous Analysis and Design

Simcode	Simulation Code
SLP	Sequential Linear Programming
SQP	Sequential Quadratic Programming
SSO	Subspace Optimization
STEP	Standard for Exchange of Product Data
SUA	System Uncertainty Analysis
SUMT	Sequential Unconstrained Minimization Technique
TASOP	Transport Aircraft Synthesis and Optimization Program
Tcl	Tool Command Language
T/W	Thrust to Weight Ratio
UAV	Unmanned Aerial Vehicle
VCM	Variable Complexity Modelling
W/S	Wing Loading (lb/ft^2)

GLOSSARY OF TERMS

- Active constraint:* an equality constraint or an exactly satisfied inequality constraint at a given design point.
- Design point:* a single, corresponding set of values for all design variables.
- Design space:* a multidimensional design variable space, in $(\tilde{x})$, that encompasses all infeasible and feasible design points. Design spaces are often described by the types of design variables they include (continuous, discrete, integer, binary, or mixed).
- Design variables:* a vector, $(\tilde{x})$, of independent and dependent parameters that represent design data. Design data that is not subject to change are referred to as design constants.
- Equality constraints:* a set of functions, $h_j(\tilde{x})$, that represent the characteristics of and limitations imposed upon a design. These constraints may be linear or nonlinear in nature.
- Feasibility:* a design point is feasible if it satisfies all of the design constraints. The measure of merit of the design point is given by the value of the objective function. The goal is to find the feasible point with the smallest objective function value (in a minimization problem).
- Global optimum:* a feasible design point that has the smallest objective function value compared to all other feasible design points in the design space.

Inactive constraint: an inequality constraint that is satisfied without directly limiting the value of a design variable at a given design point.

Inequality constraints: a set of functions, $g_i(\tilde{x})$, that represent the characteristics of and limitations imposed upon a design. These constraints may be linear or nonlinear in nature.

Local optimum: a feasible design point that has the smallest objective function value compared to other feasible design points in a local region of the design space.

Objective function: a scalar function, $f(\tilde{x})$, of design variables that are to be minimized. To convert objectives that should be maximized to a minimization problem, multiply the maximization objective function by negative one. The objective function may be linear or nonlinear, and may be composed of a set of individual objective functions, which are often contradictory.

Optimization Algorithm: a series of steps operating upon a mathematical model of a system in an attempt to find an optimal solution to a given problem.

Optimization Framework: a problem solving environment that can support the implementation and execution of MDO applications and can provide a set of database, interface, communication, optimization, visualization, and debugging services commonly required in applications of this type.

Optimization Procedure: an optimization procedure includes modelling and approximations, a decomposition method, and an algorithm to solve a problem, which may require the use of sensitivity analyses in the process.

Side constraints: upper, x_k^U , and lower, x_k^L , bounds limiting the values of the design variables. Discrete, integer, and binary values of design variables are also considered side constraints, but cannot be represented by the inequality formulation shown in Section 2.

1. Introduction

Over the last two decades, multidisciplinary design optimization has developed into a discipline unto itself. The MDO methodology has the potential to decrease the amount of time spent in the conceptual and preliminary design phases of a project while increasing the number of concepts evaluated and the level of detail to which each is examined. MDO is, in fact, necessary to create advanced designs that are competitive in terms of performance, manufacturability, serviceability, and cost effectiveness.

Without attempting a comprehensive survey, the literature review of this thesis summarizes the effort to date on the subject of MDO of aerospace design, with a distinct focus on the application to conceptual aircraft design and related software. An approach to organization of content, augmented from the survey by Sobieszczanski-Sobieski and Haftka [1], is taken to introduce the concept and components of MDO.

1.1 Multidisciplinary Design Optimization

Multidisciplinary design optimization is defined as a methodology for the design of complex engineering systems and subsystems that coherently exploits the synergism of mutually interacting phenomena [2]. The objective for such systems is global in nature: to provide the best design considering all of the constraints and relationships in the system. Multidisciplinary design analysis (MDA) evaluates the dependent variables associated with a given design point from discrete disciplines, and is an integral part of MDO. The

optimization of the structural design of an aircraft wing to prevent flutter [1] illustrates the difference between MDA and MDO. MDA of the wing will reveal if a given design is prone to flutter; MDO will alter the structural design to prevent flutter.

1.2 Design Development

Essentially, any process of design consists of a sequence of steps that can be broken down into conceptual, preliminary, and detailed phases. Production and product support follow. The linear nature of traditional design processes can be seen in Figure 1- 1 [3].

Figure 1- 1: Design Development Phases

Conceptual design begins after receipt of a set of mission requirements, from which the system requirements arise. Many simple concepts are examined, trade-off studies

performed, and a single configuration is chosen for preliminary design. Preliminary design involves more sophisticated analysis than the conceptual level to evaluate the configuration. Detailed design involves refinement at the subsystem and component levels that completes the process, resulting in a product that is ready for the production phase. In Figure 1- 2 [3], the relative knowledge about the design and design freedom are plotted throughout the various design phases, as well as the contributions from typical aerospace disciplines in each phase.

Figure 1- 2: Design Knowledge and Freedom

Design freedom decreases as knowledge about the design increases. Design variables that are set early on need to be retained as free variables much longer into the process to be able to find an optimal solution, while more information is required at earlier stages to better analyze a concept. MDO seeks to increase both freedom and knowledge (refer to Figure 1- 3 [3]) to obtain an optimal design. Efficient MDO balances contributions from

all disciplines, shortens the overall design development time, and increases the number of optimized alternative designs.

Figure 1- 3: Effect of MDO on Design Phases

To date, MDO methods have been applied to individual design phases. Due to limitations of computing hardware and system integration, expectation of optimization of the entire design process is premature at this time.

Conceptual design optimization looks at parametric configurations that describe the overall design. The optimizer replaces the traditional trade-off studies for configuration comparison, and selects a feasible, optimized concept for further definition. Examples of conceptual design optimization software are discussed in Section 6. Preliminary design optimization analyzes the baseline configuration in greater depth. For example, Pratt and Whitney Canada [4] are developing a preliminary multidisciplinary design optimization

tool for gas turbine engines using an optimization framework called iSIGHT. The optimization framework links tools from various disciplines, automates the data transfer between the tools, and automates the optimization process. This is very similar to a conceptual optimizer, except that the discipline specific software uses more complex analysis techniques to optimize a conceptual design. Detailed design optimization typically involves a subsystem or part. Much of the topology and performance of the design is already pinned down; the details are simply refined.

1.3 From Conceptual Design Iteration to Automation

Conceptual design is usually not a predetermined and strictly rational process [5]. Traditionally, a designer begins with the mission requirements supplied by a customer, an educated guess as to what future customers might need, or a new technological concept. Based on the designer's experience, a rough sketch and sizing estimate are developed. In the early stages, heuristics or "rules-of-thumb" are used to modify the layout of the design. Disciplinary analyses, in the form of manual computation, are performed to flesh out the design. Due to strong interaction between the disciplines, the resulting problem does not have a closed-form solution and requires iteration to solve [6]. Iteration ends when suitable convergence (specified by the designer) is obtained, and a baseline design is selected to be developed in detail.

Parametric or "trade-off" studies are assessments of the effect of design variables on the response of the system. One or two parameters are varied, while others are held constant,

and the effect of the change on the system as a whole is considered. Results are typically plotted on 2- or 3-D graphs or carpet plots, from which the designer makes decisions that guide system synthesis. Parametric studies tend to be time consuming, and results depend strongly on the range over which the parameters were varied, which parameters were held constant, and the values at which the parameters were held constant [7].

Conventional preliminary design optimization is a chain of sub-optimizations of individual disciplines via numerical gradient methods. Trade-off studies are conducted on individual components or subsystems (e.g. wing, engine, fuel system) using tailored analysis codes. The results of each study are then passed to the other design disciplines, which in turn analyze the impact of the trade-off on their area of responsibility. If the affected disciplines approve of the trade, their components or subsystems are modified, and the change is distributed to the remaining design disciplines for evaluation. This two-dimensional process of change and evaluation is repeated until the design converges on a mutually satisfactory solution. Unfortunately, this path along local optima can often miss a globally optimum solution [8]. For instance, the local aerodynamic optimum for minimum drag of a wing produces an elliptic pressure, and hence, load distribution along the span. A more global optimum is found when both aerodynamic and structural considerations are taken into account. Minimizing wing weight and drag produces a wing that is more heavily loaded at the root and lighter at the tip than the elliptic case, but the reduction in loading reduces the wing weight significantly enough to outweigh the drag penalty. Including manufacturing considerations in the optimization would further alter the optimal solution.

The goal of automated MDO is to achieve a near-global optimum solution through numerical optimization of a system. The level of accuracy, or fidelity, and amount of interdisciplinary coupling determines the phase to which automated MDO can be applied. As designs grow more complex, the size of the design space increases, more disciplines become involved, and more complex analyses are required to determine the response of the system.

1.4 The Nature of the Problem

Most MDO problems include a large number of design parameters and constraints, with complex coupling. The design space may be a mix of continuous, discrete, integer, or binary variables. Traditional optimization methods are ill equipped to handle mixed-variable design space in high dimensions [9], but progress has been made using artificial intelligence techniques in recent years. Optimization algorithms also run into scaling difficulty when the design parameter and constraint values are more than a few orders of magnitude apart.

Design constraints and objectives may be nonlinear in nature, increasing the difficulty of the optimization procedure. Nonlinear optimization methods provide no guarantee of a global optimum solution, although local optima can be found and declared “good design solutions” after exhaustive searches of the design space.

The shape of the design space affects the ease of problem solution. Convex design spaces allow traditional search methods within the space to be used, as they are guaranteed to lead to optimal solutions. Non-convex and disjoint spaces require stochastic or mathematical programming techniques that are more advanced. Figure 1- 4 illustrates the differences in the shape of the design space in a 2-D problem.

Figure 1- 4: Design Space Shapes

Results from the mathematical modelling of real systems depend on the level of accuracy, or fidelity, of the model. Analytical or numerical system models may be approximate or may not even exist for a particular problem (e.g. manufacturing, life-cycle costs). Optimization of an approximate or incomplete model may lead to a design that may not meet the design requirements, or even work in the real world. Hence, the designer is not only a necessary, but also a key component of the optimization process - which creates additional organizational and software complexity.

As a whole, the general MDO problem is quite complex. However, automated MDO does have significant advantages. Labour, time, and human error are minimized, resulting in

reduced costs and overall development time. More realistic geometries emerge earlier in design phases and designs are better optimized, thus mitigating risk when entering later design phases.

1.5 Issues with Early Automated MDO

Early applications of automated optimization were constrained by hardware limitations. As the computational power of personal computers and parallel processing networks increases, the cost of and time required for running computations decreases. Optimization tools are becoming more generally accessible, and more accurate (and, hence, complex) methods can be used.

Another key problem has been how to set up an MDO problem to take advantage of the potential of parallel tasking and distributed computing environments. The concurrent execution of a series of tasks requires an efficient breakdown of the problem into independent sub-problems. Several decomposition methods have been developed in recent years that can take advantage of multiple processors and distributed networks (see Section 3.5).

Drawbacks of software that was initially developed include size, complexity, and expansion, maintenance, and modification difficulties. Knowledge of the optimization code at a detailed level is often required to use or adapt the software to new problems

[10]. New software is required that can adapt to changes in design problems and is easy to use.

Automated MDO is not meant to replace the designer; it is simply another tool in the designer's toolbox. New software must address the following issues of the designer as a key component in the process:

- Interaction; the designer is a source of input, and must be able to interpret and use the output. Solutions must be in a meaningful format, understandable by the designer.
- Visibility; hidden or obscure decisions by a program confuse designers and they may not trust or may reject results.
- Flexibility; the ability must exist for the designer to interrupt or alter the optimization procedure.
- Creativity; software programs are unable to explore unconventional configurations or design fixes.

1.6 Multidisciplinary Optimization of Conceptual Aircraft Design

The design development of an aircraft has evolved naturally into a multidisciplinary-style process, where tasks are divided among discrete disciplines such as aerodynamics, structures, propulsion, and avionics. Complex and, often, conflicting design drivers from each of the disciplines influence the design process and can often lead to a sub-optimal

design. MDO methodology has the potential to balance the design decisions and produce a more globally optimal aircraft design. Since 95% of the life cycle cost of an aircraft is locked in during the conceptual, preliminary, and detailed design phases [11], a more optimal design arrived at earlier in the process will save money over the entire life span of the aircraft.

The conceptual phase of aircraft design is particularly well suited to the application of multidisciplinary optimization. A larger number of aircraft configurations can be studied in a shorter period of time, and to a greater level of detail, than can be considered using manual computation. The analyses performed at the conceptual level are perhaps the easiest to model, and a better design resulting from the conceptual phase reduces the amount of analysis, testing, and hence, cost in the preliminary design phase.

There exists a need for a software tool for conceptual design optimization of aircraft that addresses the issues that have arisen in the past with respect to automated MDO. The scope of the thesis literature review encompasses the basic components of an MDO problem; the optimization procedure; specific issues with respect to aircraft design in an MDO environment; a history of aircraft synthesis and conceptual optimization software; and requirements for new software. The structure for the MDO project at Carleton University arises from the design requirements for new software. A description of the project plan is provided, with an emphasis on the author's contribution to the work. The author's contribution is based upon the following objectives:

- To provide a comprehensive introduction to MDO, as it is a new field of research within the Mechanical and Aerospace Engineering Department at Carleton University.
- To establish a flexible software structure for performing design optimization and laying the foundation for future work within the same.
- To include conceptual design in the optimization process while including the possibility of preliminary design as well.
- To model the conceptual design process taken by designers in industry.
- To establish a generic aircraft geometric configuration that is flexible and reusable in later stages of design work.

To that end, documentation of the software structure developed and results of the conceptual design optimization process are presented with conclusions and recommendations for future work on the project. Detailed documentation of the problem formulation, assumptions, software code, and installation instructions is available in [12] as a supporting document to this thesis.¹

¹ Reference [12] is available in either electronic or hard copy format from Professor Paul V. Straznicky.

2. Problem Formulation

In the most general case, a constrained multi-objective optimization problem can be expressed in the following classical formulation:

$$\begin{aligned} \text{MINIMIZE} \quad & f(\tilde{x}) \\ \text{SUBJECT TO : } & g_i(\tilde{x}) \leq 0 \\ & h_j(\tilde{x}) = 0 \\ & x_k^L \leq x_k \leq x_k^U \end{aligned} \tag{2-1}$$

where $f(\tilde{x})$ is the objective function subject to a number of inequality and equality constraints. Each component of the problem formulation will now be discussed in detail.

2.1 The Objective Function $f(\tilde{x})$

Objective functions provide a means of comparing dissimilar design points through a single numerical value and they can be formed in a variety of ways. Objective functions may be linear or nonlinear functions of the design variable of interest. A single measure of merit may be minimized, or a multiobjective approach can be taken. The function is not limited to a simple summation of weighted measures of merit [13]. Dovi and Wrenn [14] compare the performance of three nonlinear objective function formulations. Messac [15] lists seven different linear and nonlinear formulations. Objective function formulations can significantly affect the optimal solution. Often, the objective function is

modified as designs emerge from an optimization procedure to address the shortcomings of the formulation.

There are also several alternative methods to the traditional objective function formulation, such as penalty functions and physical programming techniques.

2.1.1 Penalty Functions

Constrained optimization problems are more difficult to solve than unconstrained optimization problems. As a result, the Sequential Unconstrained Minimization Technique (SUMT) method was developed to convert constrained into unconstrained optimization using penalty functions. The constraints are represented by penalty functions added to the objective function. An exterior penalty function is set to zero inside the feasible region, and increases in value beyond the constraint boundary. Thus, to minimize the objective function, the system is driven to solutions that either satisfy the constraints or minimize the amount of constraint violation. Barrier functions, or interior penalty functions, add to the objective to ensure that a feasible solution never becomes infeasible. To minimize barrier functions, the equality constraints are converted into functions that tend to zero, drawing the optimal solution towards points that satisfy the equalities. Inequality constraints are converted to functions that grow large as a solution approaches the constraint, causing the optimizer to shy away from the boundaries, preventing the solution from becoming infeasible.

2.1.2 Physical Programming

In conventional objective functions, arbitrary weightings are used to convert design variables into common units and express a degree of influence on the optimal solution. The weightings are devoid of physical significance and often difficult to determine. Physical programming obviates the need to adjust weightings manually by mapping the variables into a common, non-dimensional space. The designer is able to express degrees of desirability explicitly for every design variable in terms of numerical ranges. Class functions are applied to the numerical ranges after the variables are mapped into the objective function space [16]. The aggregate objective function is then formed from the applicable classes. Messac and Wilson [16], and Messac and Chen [17] provide further information on design variable classifications, class functions and the aggregate objective function.

2.2 The Constraints $g_i(\tilde{x})$, $h_j(\tilde{x})$

Three types of constraints that can be applied to an optimization problem are inequality, equality, and side constraints. Inequality and equality constraints are functions of the design variables, and can be linear or non-linear in nature. Inequality constraints may be greater than ($>$), less than ($<$), greater than or equal to ($\geq$), or less than or equal to ($\leq$). However, the constraints must be converted to the form shown for $g_i(\tilde{x})$ in Equation 2-1 for use with most constrained optimization algorithms. An example of an inequality constraint is illustrated by the requirement for a jet engine aircraft balanced field length to

be less than or equal to the field length available at the smallest airport out of which the aircraft will be operating.

Side constraints are simply bounds on individual design variables that restrict the allowed values for feasible solutions. For example, the number of engines on an aircraft is limited to a non-negative integer value such that

$$\text{Number of Engines} \in \{1, 2, 3, \dots, n\} \quad (2-2)$$

where n is a reasonable upper limit on the total number of engines, which, on conventional aircraft is typically limited to four.

2.3 The Initial Design Point

Optimization is an iterative process, and most algorithms require an initial design point from the designer to be able to start the optimization process. The initial point can be an infeasible guess for robust algorithms; others require a feasible design near the optimum for convergence. As Chinneck [18] states: “To successfully optimize a problem, start at the optimum!”

A “cold start” is an initial design point for an optimization that is not based on knowledge of the design space. A “warm start” uses data from past optimizations or designs on related problems to create an initial design point. “Hot starts” use the optimal solution

from a similar optimization that is known to be near the optimal solution of the problem at hand.

The quality of the initial design point for optimization performance underscores the need for databases of similar optimizations and knowledge of the design space.

2.4 The Optimal Solution

The optimal solution to an optimization problem is the best design point within the feasible design space. The “best” design is defined as the design point with the smallest overall objective function value: the global optimum. Depending on the shape of the design space and the algorithm in use, the globally optimal solution can be difficult to locate. Optimization algorithms, especially gradient-based methods, often become trapped in a local optimum, or are unable to find any optima at all. On the other hand, some algorithms find multiple optimal solutions, a set of design points known as Pareto optimal solutions.

2.4.1 Global and Local Optima

The difference between a local and global optimum are easily illustrated using the two-dimensional design problem shown Figure 2- 1. The x- and y-axes are design variables, while the z-axis represents the objective function value. In a minimization problem, local optima are found in the “valleys” of the design space and the global optimum is located in the “valley” with the smallest z-value.

Figure 2- 1: Global and Local Optima

2.4.2 Pareto Optimal Solutions

Pareto optimal solutions are feasible solutions that arise from conflicting multiobjective considerations with equivalent objective function performance. A solution is said to dominate another when it is at least as good with respect to every objective and is strictly better with respect to at least one objective. Pareto optimal solutions are the set of all undominated feasible solutions [19]; thus, there may be more than one globally optimal solution to an optimization problem.² In Figure 2- 1, a Pareto optimal solution set would appear as two or more “valleys” with equal depths and the lowest z-value in the design space.

² A good example of a Pareto optimal solution is found in a game known as the prisoner's dilemma. Two prisoners are held for a crime that both committed, and each wishes to minimize their jail sentence. If both people confess to the crime (not knowing what the other will do), they both get five years of jail time. If only one person confesses, he gets a lighter sentence (two years) for cooperation and the partner gets a longer sentence (ten years) with a conviction based on solid evidence. If neither person confesses, it is more difficult for the state to present the case and the expected sentences will only be three years each. In this game, both prisoners will most likely confess to try to minimize their own sentence, where as the Pareto optimal solution would be for neither to confess.

From http://www.digitaleconomist.com/game_theory.html

3. MDO Components

The conceptual components of MDO, as defined by Sobieszczanski-Sobieski [1,20], include mathematical modelling, approximation techniques, optimization procedures, system sensitivity analyses, optimization procedures with decomposition, and human interface. Some categories were renamed within the thesis to more accurately reflect the concepts, and an additional category was added, “Dealing with Uncertainty,” since non-deterministic systems require a different approach to problem formulation. A brief definition of each component and the most popular methods of execution are provided in the following sections.

3.1 Mathematical Modelling

Mathematical models are used to represent the physical characteristics, behaviour, and phenomena associated with an engineering design. The models enable prediction of the characteristics of the system when certain quantities are known.

Mathematical models in MDO of engineering systems are typically implemented as assemblages of software modules. Each module represents an aspect of the system [1], such as geometry, performance, or cost estimation, and possesses a set of design variables and constraints (analyses). The relationship between the design variables in different modules is known as coupling. The strength of the coupling determines the amount of data transfer required between modules.

Depending on the history of the system (age, stage of development, design phase), the level of accuracy or “fidelity” of the mathematical model varies. Models that are more complex tend to exist for systems that are more complex, well developed, and in the later stages of design development. Software modules that use complex models are more accurate but are also more computationally expensive. This is where approximation techniques become important.

3.2 Approximation Techniques

Approximation techniques map the functional relationships between the design variables and the response of the system over a range of variation in design variables [9]. Approximation techniques are used in lieu of more exact, computationally expensive models. Caution is required when using an approximation technique in an optimization framework. Optimization algorithms do not understand that certain considerations have been simplified or left out. The algorithms will exploit weaknesses in the approximation of the analysis if it leads to a better solution [21]. Testing is critical to the success of an approximation technique.

Approximation techniques can be divided into four classes: generic approximations, corrected approximations, reduced order methods, and quasi-approximation techniques. Figure 3- 1 [2] shows how the first three approximation types interact with an optimizer and further details are provided in the following subsections.

Figure 3- 1: Approximation Processes

3.2.1 Generic Approximations

Generic approximations, either at a local or global level, provide smooth, simple, explicit analytical expressions, with little computational expense. The optimizer can generate these approximations automatically, or they can be created off-line by disciplinary experts [2].

Types of generic approximations used include [22]:

- Response Surfaces (polynomial fits using)
 - Taylor Series Expansion
 - Linear Functions
 - Quadratic Functions
 - Higher Order Polynomials
 - Two-Parameter Interaction Terms

- Neural Networks
 - Backpropagation
 - Counterpropagation

Generic approximations are typically constructed using a stochastic search technique to generate sample data. The accuracy of these methods is highly dependent on the amount of data used to create them. Local augmentation to the amount of sample data is crucial to increase the fidelity of the approximation in critical design regions.

3.2.2 Corrected Approximations

Corrected approximation techniques use low fidelity disciplinary codes that have been corrected using high fidelity codes or experimental results. The caveat on such techniques is to ensure that all phenomena are appropriately modelled so that the high fidelity information can be utilized effectively [2].

Variable complexity modelling (VCM) uses a linear or multiplicative scaling factor provided by the high fidelity code to correct the approximation model [23]. The scaling factor is periodically updated to maintain the validity of the approximation.

An alternate technique interchanges the use of an approximation with that of a full analysis. The optimum of the approximate problem is found; a full analysis is completed for the optimum point; the approximation is updated, and the process repeats [1].

3.2.3 Reduced Order Approximations

Reduced order methods express the results of high fidelity numerical analyses in simple analytic expressions. Examples of reduced order methods include state space representations of dynamic models and associated rational function approximations [2].

3.2.4 Quasi-Approximation Techniques

Design variable linking and basis reduction may also be considered approximation techniques [24], as they introduce simplified assumptions into the design space. Design variable linking specifies direct relationships between different variables (e.g. $x=3y_1+y_2$), which occur often in geometric representations. Basis reduction uses a weighted sum of candidate designs, where the weightings are the independent design variables instead of the candidate design parameters. For example, the ordinates, $\tilde{y}$, of an airfoil shape can be defined by one hundred independent design variables, $\tilde{y} = [y_1, \dots, y_{100}]$. Four candidate shapes are selected that span the design space, as shown in Figure 3- 2.

Figure 3- 2: An Example of Basis Reduction

The optimal shape is given by a weighted, vectorized sum of each candidate design, where the weightings, a_j , on each vector become the design variables used in the optimization; thus reducing the number of independent design variables from one hundred to four.

3.3 Optimization Algorithms

Given an initial design point, an optimization algorithm seeks new solutions through improvement in the value of the objective function. The defining characteristic of an algorithm is the way in which it finds a new solution. Five classes of optimization algorithms will be discussed, all of which can be applied to nonlinear optimization problems. Algorithms specific to linear programming will not be discussed, as most aerospace design optimization problems are nonlinear in nature.

The comparison of performance of different algorithms is difficult, as some algorithms can only be applied to certain types of problems, while others may find equally good designs in different parts of the design space. References [25], [26], and [27] compare the performance of some nonlinear optimization algorithms.

3.3.1 Enumeration

Enumeration consists of trying all possible solutions, and simply choosing the one with the best value of the objective function. For a problem with only one or two variables that are limited to a few discrete values, the computational demands are tolerable. In higher

dimensions with continuous variables, enumeration is extremely inefficient. Even a random search is more efficient than enumeration.

3.3.2 Stochastic Search Methods

Stochastic techniques are used to generate both search directions and initial design points for optimization algorithms. Random search methods use a simple iterative update strategy,

$$\tilde{x}^{i+1} = \tilde{x}^i + \alpha S^i \quad (3-1)$$

where S is the search direction, and α is the step size to be taken in the search direction. The step size is started large, and search directions are accepted only if they result in an improved value of the objective function. The algorithm stops when a new S and reduced α does not result in improved design [9]. These searches tend to be inefficient. Stochastic sampling is used more effectively to generate points in the design space when coupled with a highly exploitive search technique to select the best design.

Monte Carlo Simulation (MCS) is an unbiased, exhaustive, computationally intensive method used to sample the entire design space. MCS is similar to enumeration in that a very large number of design points are evaluated; however, MCS randomly selects the values of variables, based on given probability distributions, instead of evaluating every possible value. MCS is more efficient than enumeration and is often used as a crosscheck for other optimization algorithms.

Design of Experiments (DOE) is an unbiased, structured method of generating variables in an iterative process to obtain solutions throughout the design space. Designs are generated to achieve the maximum knowledge of the design space with a minimum number of system level evaluations. Taguchi Methods, Central Composite Designs, and saturated D-Optimal Designs [28] are all refined DOE methods.

Simulated annealing is a computational method based on an analogy to the annealing of metals, which, when cooled slowly, attain a state of minimum energy. The algorithm is initialized to an arbitrary “temperature” value, where a large number of randomly sampled design points are evaluated. The energy state of a design point is given by the value of the objective function. The design point with the lowest energy state is accepted as the solution estimate for the next iteration. Acceptance of a point with a higher energy state than the current design is random, with a probability that decreases exponentially with temperature [26]. The temperature is reduced slowly over a number of iterations according to a set cooling rate, and the energy states of design points near the current solution estimate are calculated. The algorithm stops when an arbitrary temperature is reached, changes in the evaluated energy states are sufficiently small, or limits are imposed on the number of iterations. Simulated annealing is a robust, unspecialized process, and is especially useful when no other algorithm lends itself to a particular optimization problem.

Genetic algorithms (GA's) are based on the paradigm of Darwinian evolution. Every design point is coded into a concatenation of binary representations of the independent design variables similar to strings of genetic code. A population of designs is formed from a stochastic sampling of the design space, and evolved using reproduction, mutation, and crossover. "Survival of the fittest" is reflected in the value of the objective function, which is a numerical measure of the "fitness" of a design. The fittest designs are then selected for reproduction, for which there are several crossover schemes available. Mutation introduces new elements into the design using random number generation. GA's are easy to code, can solve optimization problems involving mixed, continuous, and discrete design variables, and are less likely to be trapped in local optima. However, GA's are not as efficient as gradient-based search methods near an optimal solution [29].

3.3.3 Optimality Criteria

The indirect method of optimization, otherwise known as the method of Lagrange, seeks local optima by solving the set of equations resulting from setting the gradient of the objective function to zero (for an unconstrained optimization).

$$\nabla f(\tilde{x}) = 0 \quad (3-2)$$

Constrained optimizations use Lagrange multipliers (λ) to incorporate the active constraint information into the gradient, where the constraints must be equalities.

$$\nabla f(\tilde{x}^*) + \sum_{i=1}^m \lambda_i \nabla g_i(\tilde{x}^*) + \sum_{j=m+1}^{m+n} \lambda_j \nabla h_{j-m}(\tilde{x}^*) = 0 \quad (3-3)$$

The * in Equation 3-3 denotes an augmented set of design variables, including those required to convert inequality to equality constraints. It should be noted that optimality criteria are based on the governing equations of the system, and are not applicable when the analysis combines governing equations of very different phenomena [3].

By solving Equation 3-3 as a system of simultaneous equations in both design variables and Lagrange multipliers, a feasible design point and the corresponding multiplier values can be computed. The design point is guaranteed to be a local optimum if it also satisfies the original constraints plus the following conditions:

$$\begin{aligned} \lambda_i g_i(\tilde{x}^*) &= 0 \\ \lambda_j h_j(\tilde{x}^*) &= 0 \\ \lambda_{i,j} &\leq 0 \end{aligned} \quad (3-4)$$

Altogether, the conditions are known as the Karush-Kuhn-Tucker (KKT) conditions, and are the only way to guarantee local optimality (global optimality is never guaranteed).

3.3.4 Gradient-Based Search Methods

Gradient-based methods are a direct method of optimization. Optimal solutions are sought by starting with the objective function and using its gradient to guide the search.

The first and, sometimes, second derivatives of the objective function with respect to the design variables are calculated using a form of sensitivity analysis.

Gradient-based optimization algorithms are often known as “hill-climbing” methods. Similar to the fact that fastest way to the top of hill, or to the bottom of a valley, is the steepest path, the fastest way to reach a local maxima or minima is in the direction of the largest value of the function gradient. Thus, the simplest gradient method for a minimization problem is the path of steepest descent. Given an initial point, the objective function, f , and its gradient, ∇f , are calculated.

$$\nabla f = \left(\frac{\partial f}{\partial \tilde{x}_1}, \dots, \frac{\partial f}{\partial \tilde{x}_k} \right) \quad (3-5)$$

The new design point is calculated from a simple, iterative, update strategy,

$$\tilde{x}^{i+1} = \tilde{x}^i - \alpha * \nabla f \quad (3-6)$$

where i is the iteration number, and the negative of the gradient is the search direction that most improves the value of the objective function. The search direction is multiplied by α , a scalar step size, that is arbitrarily chosen. The step size is then increased until the new objective function value is no longer an improvement over the old value. A new gradient is calculated and the process repeats [7].

Gradient methods can be augmented in a number of ways, including variations in the update strategy or step function (e.g. trust region methods) and using the second derivative of the objective function (the Hessian) to improve the estimate of the search direction. Advanced gradient methods include:

- Variable Metric or Quasi-Newton Method
- Generalized Reduced Gradient (GRG) Method
- Powell's Method
- Conjugate Gradient Method
- Method of Feasible Directions
- Modified Method of Feasible Directions (MMFD)
- Sequential Linear or Quadratic Programming (SLP, SQP)

Gradient-based searches are limited to unconstrained nonlinear problems whose objective function is smooth and differentiable, and are typically used in conjunction with penalty functions for constrained problems. Non-smooth functions, termed “noisy,” can cause severe difficulties in the search.

3.3.5 Pattern Search Methods

Pattern search algorithms rely on knowledge of the design space, rather than random sampling or gradient information. The search direction is chosen using patterns in the values of the design variables, i.e. binary, integer, or set increments of values. Examples of pattern search methods include:

- Branch and Bound Method [18]
- Hooke and Jeeves Pattern Search [18]
- Nonlinear Simplex [21]
- Parallel Direction Search [21]

3.4 System Sensitivity Analysis

Sensitivity analysis consists of computing the derivatives of one or more output quantities with respect to one or several independent input variables [30]. There are two aspects of an MDO problem with respect to sensitivity analysis. One is the sensitivity of the response (dependent variables) to the design variables (independent variables); the other is the sensitivity of the optimal solution to all problem parameters. Sensitivity analysis techniques can be applied to compute either aspect, but the Global Sensitivity Equations (GSE's) were developed to address the latter concern specifically; refer to Section 3.4.2 for more information.

Sensitivity analyses are useful in establishing search directions for gradient-based optimization algorithms, for constructing approximate analysis models, and for performing trade-off studies. They also play an important role in decomposition-based design [31].

Optimal sensitivity analysis, or post-optimality analysis, is an equivalent analysis performed after the initial optimization problem is solved. Optimal sensitivity

information is used to guide parametric studies for preliminary design and reducing the need for sensitivity analysis on a re-optimized sub-problem [32]. Additional references on this topic include [33,34], and a test case from [35].

3.4.1 Sensitivity Analysis Techniques

There are four distinct techniques available to perform a sensitivity analysis: finite differencing, complex variable programming, automatic differentiation, and the analytical approach.

Finite differencing consists of a truncated Taylor series approximation of the function derivative $f'(x)$ expanded about the point x . The forward-difference gives a first order estimate,

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \quad (3-7)$$

while the central-difference gives a second order estimate of the derivative.

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad (3-8)$$

The truncation error, $O(h^n)$, indicates the order of approximation, and h is the step size.

Finite differencing is neither very accurate nor efficient. Accuracy is step-size dependent and subtractive cancellation errors dominate at very small step sizes. Computational cost increases linearly with the number of design variables [30]. The biggest advantage lies in the fact that the procedure is easy to implement, although special consideration is required for nonlinear analysis.

Complex variable programming, or complex-step derivative approximation, uses complex variables to develop a simple expression for the first derivative of a function. The formula can also be derived using an imaginary step, ih , in the Taylor series expansion.

$$\frac{\partial f}{\partial x} = \frac{\text{Im}[f(x + ih)]}{h} \quad (3-9)$$

The approximation is accurate, robust, easy to implement, and has a reasonable computational cost [30]. Unlike finite differencing, it is not subject to subtractive cancellation errors.

Automatic differentiation is based on the systematic application of the chain rule to symbolic expressions in existing computer programs. Many tools for automatic differentiation have been created in a number of computer languages (e.g. FORTRAN, C/C++). Automatic differentiation is as accurate as the analytical approach, and potentially easier to implement.

The analytical approach applies direct differentiation of the governing equations to determine the sensitivity of each variable to a change in another variable. It is the most accurate and efficient method for computing sensitivity information. There are two methods: the direct method, which generates the sensitivity derivatives as unknowns; and the adjoint (or indirect) method, which computes the derivatives of a selected subset of the unknowns. The rule-of-thumb is that if the number of design variables (input) is greater than the functions of interest (output), the adjoint method is more efficient than the direct approach, and vice versa [30]. The analytical approach can be generalized to a number of coupled subsystems using the global sensitivity equations.

3.4.2 Global Sensitivity Equations

Olds [7] provides a summary of Sobieszczanski-Sobieski's work [36,37] on the development of the GSE, using a three-part system example. A complex, coupled system can be thought of as a mathematical function that, for a converged solution, generates a set of output values ($\tilde{y}$) for a given set of input values ($\tilde{x}$). Therefore, $\tilde{y}$ is a function of $\tilde{x}$, and can be written as:

$$F(\tilde{x}, \tilde{y}) = 0 \quad (3-10)$$

The output variables are generally used in the evaluation of the objective function and design constraints. Assuming that the system output vector can be partitioned into distinct discipline outputs,

$$\tilde{y} = (\tilde{y}_1, \tilde{y}_2, \tilde{y}_3)^T \quad (3-11)$$

where three disciplines are utilized in this example (note that the results can be generalized to n subsystems). Combining Equations 3-10 and 3-11 gives Equation 3-12:

$$F(\tilde{x}, \tilde{y}_1, \tilde{y}_2, \tilde{y}_3) = 0 \quad (3-12)$$

Equation 3-12 is rewritten such that one variable is expressed as a function of the others using the implicit function theorem. (It is assumed that system decomposition has been performed so that a discipline is not a function of its own inputs.)

$$\tilde{y}_1 = F_1(\tilde{x}, \tilde{y}_2, \tilde{y}_3) \quad (3-13a)$$

$$\tilde{y}_2 = F_2(\tilde{x}, \tilde{y}_1, \tilde{y}_3) \quad (3-13b)$$

$$\tilde{y}_3 = F_3(\tilde{x}, \tilde{y}_1, \tilde{y}_2) \quad (3-13c)$$

Using Equation 3-13a as a representative example, after applying the chain rule to get the differential form:

$$d\tilde{y}_1 = \frac{\partial \tilde{y}_1}{\partial \tilde{y}_2} d\tilde{y}_2 + \frac{\partial \tilde{y}_1}{\partial \tilde{y}_3} d\tilde{y}_3 + \frac{\partial \tilde{y}_1}{\partial \tilde{x}} d\tilde{x} \quad (3-14)$$

The total derivative is given by:

$$\frac{d\tilde{y}_1}{d\tilde{x}} = \frac{\partial\tilde{y}_1}{\partial\tilde{y}_2} \frac{d\tilde{y}_2}{d\tilde{x}} + \frac{\partial\tilde{y}_1}{\partial\tilde{y}_3} \frac{d\tilde{y}_3}{d\tilde{x}} + \frac{\partial\tilde{y}_1}{\partial\tilde{x}} \quad (3-15)$$

Performing a similar process on the other two subsystems will lead to the coupled matrix equation known as the Global Sensitivity Equation for the system.

$$\begin{bmatrix} I & -\frac{\partial\tilde{y}_1}{\partial\tilde{y}_2} & -\frac{\partial\tilde{y}_1}{\partial\tilde{y}_3} \\ -\frac{\partial\tilde{y}_2}{\partial\tilde{y}_1} & I & -\frac{\partial\tilde{y}_2}{\partial\tilde{y}_3} \\ -\frac{\partial\tilde{y}_3}{\partial\tilde{y}_1} & -\frac{\partial\tilde{y}_3}{\partial\tilde{y}_2} & I \end{bmatrix} \begin{bmatrix} \frac{d\tilde{y}_1}{d\tilde{x}} \\ \frac{d\tilde{y}_2}{d\tilde{x}} \\ \frac{d\tilde{y}_3}{d\tilde{x}} \end{bmatrix} = \begin{bmatrix} \frac{\partial\tilde{y}_1}{\partial\tilde{x}} \\ \frac{\partial\tilde{y}_2}{\partial\tilde{x}} \\ \frac{\partial\tilde{y}_3}{\partial\tilde{x}} \end{bmatrix} \quad (3-16)$$

The global sensitivity matrix on the left expresses the sensitivities of each discipline to other discipline outputs. The vector on the right contains the sensitivities of each discipline to inputs while holding other influences constant. Given both these sets of values from discipline-level sensitivity analysis techniques, the global sensitivity information can be calculated. This calculation is equivalent to performing a finite difference analysis on the entire system for each design variable, but with far greater computational efficiency.

Sobieszcanski-Sobieski also defines a method for determining higher order sensitivity information in [38].

3.5 Decomposition Methods

MDO problems are typically quite large and complex. The initial formulation of the problem requires a designer to examine what design variables, constraints, and objective functions should be used to represent the system most accurately. Once the problem has been mapped out, it can be solved using an optimization algorithm. Post-processing allows the designer to see meaningful results and guide future optimization problems. However, prior to solution and post-processing, decomposition methods are often used.

The design of the design process, or “meta-design”, seeks to improve the formulation of MDO problems by decomposing the problem into familiar, industry related disciplines to minimize computational requirements and enable use of concurrent computing capabilities. In this manner, the designer and the optimization algorithm avoid tackling the entire system at once. The form of the problem is more manageable, while the important couplings among subsystems are retained. The more complex the system or analyses, the more such couplings must be accounted for, and the amount and type of coupling determines type of decomposition to use. Care must be taken by the designer to choose the right decomposition method for the problem at hand. If little is known of the coupling structure, Wagner and Papalambros [39] define a formal methodology for selecting a decomposition method suited to the problem.

The advantages of systems with and without decomposition are discussed below, as well as manual and automated methods of decomposing the problem formulation. It is

important to note that numerous variations and combinations of the subsequently described general decomposition methods exist. Balling and Sobieszczanski-Sobieski [40] provide a unified overview of problem formulation and a structured approach to describing variation.

3.5.1 All-In-One

The all-in-one approach is known by several names. The nonlinear programming community knows it as the Multi-Disciplinary Feasible (MDF), or the Fully Integrated Optimization (FIO) method. The all-in-one approach should not be confused with the All-At-Once (AAO) problem formulation, otherwise known as Simultaneous Analysis and Design (SAND). For more information, refer to [41].

The all-in-one approach is the conventional approach to solving the MDO problem. As the name suggests, it involves no system decomposition and there is no strategy for optimal design of the problem formulation. The optimization algorithm is applied to the complete set of multidisciplinary analyses to generate solutions.

The advantages of the all-in-one approach include simplicity of organization (i.e. none required) for the designer, and less chance of the optimization algorithm being caught in an artificial local optimum. Conversely, the simplification of tasks for the designer increases the complexity of the problem for the optimization algorithm, as it must deal with all design variables and constraints at once. Every step involves a system analysis that is computationally expensive for the same reason. In addition, the lack of modularity

of such an approach is disadvantageous as it disconnects the problem from the traditional industry disciplines. The all-in-one approach is, however, successful on a level to which the newer decomposition methods have yet to prove themselves.

3.5.2 Hierarchic Decomposition

Hierarchic systems are amenable to groupings of design variables and constraints such that relationships exist within a subsystem group but no relationships exist among different groups. A subsystem may be further broken down into independent subsystems in a multilevel process. An example of a hierarchic system is shown in Figure 3- 3, where the system level objectives are represented in the top “black box” and are linearly decomposed into individual subsystems. Arrows indicate data flow based on relationships between systems.

Figure 3- 3: Hierarchic System Decomposition

The analysis information flow is top-down, until the bottom level of the tree is reached, then optimization progresses from the bottom level up. For example, the outputs of

subsystem 1 are constant parameters input to subsystems 1.1, 1.2, 1.3 to perform the subsystem analyses. Subsystems 1.1, 1.2, and 1.3 are optimized independently and results from the optimization are transmitted to subsystem 1. Subsystem 1 is optimized and information is then passed back to the system level.

Hierarchic systems are rarely encountered in real applications, and, often, weak subsystem relationships are simplified or neglected to attain hierarchic decomposition. The main advantage of hierarchic decompositions is individual subsystem optimization, which simplifies the optimization problem for numerical techniques. Such optimizations are also more efficient with warm or hot starts.

3.5.3 Non-Hierarchic Decomposition

The most general form of a complex system is non-hierarchic in nature. Design variables and constraints can be grouped into general functional subsystems, but multidirectional transmission of information is required due to coupling between subsystems, as depicted in Figure 3- 4.

Figure 3- 4: Non-Hierarchic System Decomposition

The method of subspace optimization (SSO) is essentially a hierarchic decomposition structure applied to a non-hierarchic system. Subsets of the vector of the design variables are changed while the remainder is held constant [42]. After each subsystem optimization, a system analysis must be performed to update the design variables. SSO is computationally expensive and time consuming. The hierarchical modularity concept can be better applied to non-hierarchic system through the method of concurrent sub-space optimization (CSSO). First proposed by Sobieszczanski-Sobieski in [42], the CSSO method has been used successfully in [43,44] and CSSO has been implemented in commercial software and tested in [45]. Examples are developed in [46,47] using approximation methods to replace the original coordination procedure, and formal software development of the modified CSSO method is described in [48]. Essentially, for every iteration of the system optimizer, the subsystem design variables are held fixed while the system analysis computes the coupling variables, and then the coupling variables are held fixed while the subsystems are optimized for the local design variables using cumulative constraints to account for the effect of other subsystems [40]. CSSO typically employs the GSE's to guide the optimization.

Collaborative optimization (CO) utilizes compatibility constraints ($x=x'$) to ensure subsystem optimizations meet local constraints while achieving a system level optimization. At the system level, design variables that couple subsystems are set as targets. Subsystems are divided into independent analyses, with their own set of local variables and local copies of the coupling variables. Each subsystem is optimized separately to minimize the difference between the global and local values of the coupling

variables. There are two collaborative strategies for subsystem optimization: minimizing the difference between the global and local coupling variables using or minimizing the tolerance on the difference as an additional variable. Note that the difference can be expressed in a number of compatibility constraint formulations [49]:

$$\begin{aligned}
x - x' &= 0 \\
(x - x')^2 &= 0 \\
\frac{x}{x'} - 1 &= 0 \\
\ln\left(\frac{x}{x'}\right) &= 0
\end{aligned} \tag{3-17}$$

The system-level coordination optimizer then attempts to minimize the overall objective while constraining the difference between the global and local values of the coupling variables to zero. Hence, multidisciplinary compatibility is attained through convergence of subsystems optimizations. The CO method fits well with industry disciplines and is well suited to large-scale problems [41], but its performance on highly coupled systems is unproven. Kroo and Manning [50] discuss performance issues of the CO method, and examples using the CO method are developed in [51].

Bi-Level Integrated System Synthesis (BLISS) is a multilevel decomposition method in which the subsystem design variables are grouped according to discipline, and system level variables maintain the coupling between subsystems. After best guess initialization, the system level analysis is performed. The system level variables are frozen, and discrete, concurrent analyses and optimizations in the subsystem domains are performed

[52]. Optimum sensitivity data or response surfaces link subsystem to system space; a system level optimization is then performed, and the cycle repeats (see Figure 3- 5 [53]). Each subsystem may be further broken down into lower level subsystems.

Figure 3- 5: BLISS Decomposition Cycle

3.5.4 Automated Decomposition

All methods of decomposition discussed in Sections 3.5.1 to 3.5.3 require the designer to apply a particular decomposition method ad hoc. To eliminate human error and reduce the time and cost of the design cycle, the decomposition process can be automated. N-square methods using design structure matrices (DSM's), coordinate transformations

used to identify orthogonal subspaces, and hypergraph model representations have been utilized for automatic decomposition [1].

Design Manager's Aid for Intelligent Decomposition (DeMAID – NASA Langley) is designed to decompose systems that involve iteration. DeMAID is a knowledge-based software tool for minimizing feedback couplings; sequencing the design processes; grouping processes into iterative sub-cycles when feedback cannot be eliminated; decomposing sub-cycles into a hierarchical, multilevel structure; and displaying the sequence of processes in a DSM [54,55,56]. Additionally, the inclusion of a genetic algorithm allows for the optimization of sequence of design processes within an iterative sub-cycle, based on arbitrary values of process computational cost, computational time, and coupling strength. DeMAID can also make suggestions on the removal or temporary suspension of processes or couplings, based on the set strength of the coupling, to help simplify the decomposition.

3.6 Dealing With Uncertainty

Most MDO studies operate under the assumption that the system in question is deterministic. That is, the mathematical model of the system is assumed 100% correct and no uncertainty in the equations or data used to develop the constraints exists. The assumption of certainty can lead to a suboptimal design under a range of operating conditions or because of variations due to uncertainty in physical properties (e.g.

geometry, materials, manufacturing) [57]. The level of error in the modelling and analysis techniques also affects the uncertainty of the system.

A relatively new area under development is robust design with multidisciplinary objectives under uncertainty. Optimization of a robust design requires evaluation of performance variations due to system randomness, as well as co-ordinating the mean and variance of performance attributes in multiple disciplines: a complex and computationally expensive task. Du and Chen [58] describe several methods of approaching the problem; all of which are modifications of methods described previously.

3.7 Human Interface

Several issues must be examined when designing a human-to-software interface. Data management and visualization infrastructure are important to enable the designer to interact with the design process. The flexibility, functionality, aesthetics, and ease of use of the interface are equally as important.

3.7.1 Data Management

Data management includes input, output, and storage issues. The input system must be flexible enough to accommodate a wide range of problems. Software output is addressed under visualization issues. Data storage is typically achieved through a database. Ideally, a database must be able to handle large amounts of data rapidly and allow multi-site, heterogeneous operation. The database itself must also be user-friendly.

3.7.2 Visualization

The designer must be able to visualize the process of analysis and optimization to arrive at a measure of confidence and understanding of the results. Ideally, the designer would have access to all calculations performed in the analysis, all data produced during the optimization as it is being run, and an examinable database of past optimizations. The real difficulty arises out of dimensionality issues. Humans are linear thinkers, and have trouble visualizing multidimensional space (greater than three or four dimensions). To facilitate the absorption of large amounts of information, text, tables, color, graphics, animation, multimedia applications, and virtual reality environments may all be used [17].

A useful representation of the numerical solution is required to communicate to the designer how the optimizer found a design point, and how that point relates to other designs in the design space. There are two methods in use: artifact-based visualization and non-artifact based visualization. Artifact-based visualization schemes use a generic geometrical description of the artifact, representing the physical features of the design at a given point in the optimization. Non-artifact based visualization schemes can be further subdivided as shown in Figure 3- 6. The techniques are self-explanatory, and are very familiar to most designers.

Visualization is more than just a way of presenting results; it is also a solution tool [17]. It allows the designer to interact with the optimization procedure. The designer can analyze the design progress, and exit if progress proceeds along an undesirable path.

Previous solutions show trends in data and can speed the process for new, related designs. VizCraft [59] and PhysPro [17] are both successful examples of integrated visualization software.

Figure 3- 6: Visualization Techniques

3.7.3 Qualitative Aspects

A flexible design interface must integrate both qualitative and quantitative decision-making. The human mind is the leading force in the design process, not the software, and the entire design of the software must recognize this. MDO software assists in problem formulation, reduction of problem complexity, analysis, and results interpretation; it does not replace the designer. To aid the designer through the process, comprehensive help files and tutorials must be accessible, and error-handling capabilities must be included. The software must also be user-friendly, aesthetically pleasing (likely a well-organized graphical user interface, or GUI, combined seamlessly with the computational environment), and fast. Rapid turnaround is required for more simulations, better analysis, and more designer input.

3.7.4 Future Functionality

A new way in which design optimization interfaces are being explored is through the World Wide Web. Software installation issues would become nonexistent, and functions would be platform independent. Use of web-based design would also allow geographically dispersed design groups to contribute to the process.

4. Specific Issues With Respect to Conceptual Aircraft Design

The primary concern with conceptual aircraft design optimization lies in the formulation of the problem. To be useful, the problem must be defined in terms of design parameters that are commonly used in industry. Design constraints that accurately represent the behaviour of the design must be developed, while maintaining the relationships between design variables. The “measures of merit” (or MOM as defined by Raymer [60]) chosen to form the objective function drive the optimal design. If the objective function does not accurately reflect the needs of the customer, the optimized design will not be satisfactory.

An incomplete formulation results in sub-optimal designs, designs that do not satisfy mission requirements, or, at worst, a completely infeasible design. As a result, a simple, robust, problem-independent tool that takes into account all of the major issues with respect to conceptual aircraft design is required to implement multidisciplinary design optimization.

4.1 Design Variables, Constraints and Measures of Merit

To model an aircraft system accurately, the design variables, constraints, and measures of merit must be carefully chosen. Raymer [60] defines the following:

Proposed Selection Criteria: Design Variables

- 1) Represent clear physical or technical features of the design layout that can be expressed and parametrically modified by a single numerical value (e.g. wing loading), or can be clearly explained as a discrete, “either-or” choice (e.g. tail type).
- 2) Should not represent items of specific end worth or value to customers (e.g. range, speed, or cost) – such items should be incorporated into the measure of merit or constraints.
- 3) Non-trivial and calculable impact on chosen MOM (typically related to weight or range).
- 4) Global impact – i.e. affect entire design such that variables cannot be optimized independently (e.g. engine inlet duct length of commercial airliners does not usually require global optimization).
- 5) Reflect current and historical design parameters and vocabulary wherever possible.

It is important to note that the design space for an aircraft is usually mixed (i.e. continuous, discrete, integer, binary). While this makes the problem more difficult to solve, it would be inappropriate to optimize a design for 1.5 engines or an aspect ratio that only came in integer values.

Proposed Selection Criteria: Design Constraints

- 1) Represent calculable properties of the aircraft given a particular set of values of the design variables.

- 2) Must-meet numerical values; derived from commercial or military specifications, safety, operability considerations, customer desires, or overall good design practice.
- 3) Preferable if design constraints represent “firm” requirements that are unlikely to be relaxed in the face of cost/performance shortfalls.

Design constraints are not always considered firm. Often, the designer wishes to indicate a condition that is preferable, but not necessary. For example, longitudinal stability of an aircraft is not explicitly required, as an active control system can be added to the design, but it is preferable to lower costs and keeps the design simple. The use of physical programming to describe the degree of desirability of a stable aircraft would be ideal in this case (refer to Section 2.1.2).

Proposed Selection Criteria: Measures of Merit

- 1) Represent a non-trivial and calculable indication of the worth of the concept.
- 2) Materially affected by the design variables and constraints.
- 3) Clear meaning to designers and customers.
- 4) If blended (i.e. multiobjective), need clear rationale for method and factors used for blending.

There is no right way to define the “best” aircraft design. Therefore, care must be taken with the choice of objective function formulation for aircraft. If too much emphasis is placed on a particular measure of merit, a globally suboptimal design will result. Humorous examples of individual discipline design objectives taken to the extreme are

shown in Figure 4- 1 [21]. In addition, the design constraints may not leave any feasible solutions. If that is the case, use of penalty functions may be required (refer to Section 2.1.1).

Figure 4- 1: “Disciplines”

4.2 Analysis Fidelity

Bartholomew [61] defines the following set of analysis levels:

- Level 1: Empirical equations
- Level 2: Intermediate level models (e.g. beam theory, aerodynamic panel methods)
- Level 3: State-of-the-art, high fidelity models (e.g. FEA, CFD)

Prediction methods for conceptual design are level 1 equations typically derived from (semi-) empirical and statistical analysis on existing aircraft [5]. Level 2 methods are slowly being incorporated to increase the accuracy of the model. Level 3 methods are normally used in late preliminary and early detailed design optimization.

Input and processing requirements discourage the use of analysis methods that are too complex, but disciplinary experts demand high fidelity analyses for confidence in results. The idea is to achieve as high fidelity in the analysis as possible without increasing set-up and computation times beyond a few hours.

4.3 Analysis Modules

Discipline specialization has resulted from a natural decomposition of aerospace design systems along discipline boundaries. Systems that are highly coupled tend to be grouped together, and loosely coupled systems are separated for ease of visualization and computation. Because individual disciplines are so vast, specialization occurs. The discipline groups can often be represented by loosely coupled modules in design optimization software.

The principal analysis tool used during the conceptual design phase is a sizing module. After aircraft sizing, Nicolai and Carty [8] define the minimum set of disciplines required for an aircraft trade study as aerodynamics, propulsion, weights, thermodynamics, performance, stability and control, and cost. These disciplines form the basis for analysis

modules that should be included in conceptual aircraft design optimization software. Other modules to consider are manufacturing and supportability [9].

The modularity of the system should also allow for incorporation of new technologies and unconventional designs into the existing discipline-specific modules.

4.4 Geometry

The outer mould line (OML) determines the fundamental characteristics of an aircraft at the early stages of design. Ideally, various concepts are analyzed before the OML is frozen to allow as much flexibility as possible in the conceptual design phase, with a feasible, optimal design carrying over to the preliminary phase. Design optimization software can provide the designers with a tool that will analyze and optimize many designs within a short period. If the OML is effectively represented in the optimization software, less risk is associated with the resulting design(s).

It is difficult to define a set of parameters that can adequately describe the geometry of every style of aircraft, especially unconventional designs. The parametric geometry module within design optimization software must either be able to accommodate all designs, or be easily modified to include new designs. Smith and coworkers [62,63] define a parametric methodology and grid implementation.

4.5 Scaling Laws

Scaling laws are approximations to the behaviour of the system that give the designer an indirect way of accounting for variable coupling. A simplified example of a scaling law is that engine nacelle area varies roughly with the thrust level – thus the designer knows that if he uses a more powerful engine at the same weight, the nacelle area will increase proportionally. Simplified scaling laws can ease the computational burden in the optimization process, but may not reflect the “real-world” design interactions. Errors introduced by simplified scaling laws can be eliminated by participation of the designer during the iterative process [60]. However, this slows the optimization process and introduces unnecessary work.

4.6 Cost Estimation

Cost minimization is traditionally approximated by minimizing gross takeoff weight, empty weight, or fuel consumption [64]. Including the cost of an aircraft as a direct measure of merit ensures that economic feasibility information is incorporated into the design when potential impact is greatest.

The life cycle cost of an aircraft is the total cost, from design inception to retirement of the aircraft at the end of its service life. Figure 4- 2 shows how life cycles costs are broken down into components associated with the manufacturer and the customer. Note

that purchase price is not included, as it is dependent on profit margins and other business considerations that do not allow for relative aircraft design comparisons.

Figure 4- 2: Life Cycle Cost Breakdown

Each cost component has different cost drivers and most are easy to identify, such as fuel, design complexity, and incorporation of new technology. The difficulty comes when trying to define a relationship between the drivers and the cost over a long period, in an uncertain environment. A few cost estimation models have been developed, such as the

DAPCA IV (Rand Corporation) or the Weight Dependent Estimation Method (Roskam), but most are limited in scope. Broader cost models that exist within large aerospace companies, based on databases of existing aircraft, are proprietary. Johnson [64] uses a composite life cycle cost estimation method, comprised of airframe and engine acquisition costs, and direct and indirect operating costs. Cousin and Metcalfe [65] suggest that the system be sufficiently broken down into various common subsystems so subsystem-specific cost models, similar to the COSTRAN software in [66], can be applied. As well, different cost models could be available for different levels of system complexity.

4.7 Manufacturing and Assembly Issues

In an attempt to improve quality and reduce cost, manufacturing, assembly, and service issues are taken into account in the early phases of the design process. Concurrent engineering (CE), also known as integrated product and process development (IPPD), provides a framework for such considerations. CE is far easier to implement in a qualitative fashion than a quantitative one. Numerical optimization, however, requires a realistic, reliable mathematical model. In the past, both manufacturing and costs models were inadequate for MDO [3], but research focus is shifting to develop the required models.

A manufacturing and assembly model must consider the cost of raw materials, tooling of individual components, fabrication processes, jigging, and assembly (inclusive of labour).

Other factors that arise from manufacturing and assembly are scrap rate, damage tolerance considerations, repair costs, environmental factors based on the toxicity of the materials used, and certification issues [67].

5. Aircraft Synthesis Software

While aircraft synthesis software cannot perform multidisciplinary design optimization, it does provide the foundation for analysis modules that could be included in an optimization package. Listed here are only those codes commercially or publicly available, and published in literature.

A number of the software packages are labelled as “preliminary design” systems, but most fall under the scope of the more recent definition of multidisciplinary conceptual design optimization.

5.1 AAA (University of Kansas)

The Advanced Aircraft Analysis (AAA) program is a user-controlled analysis tool for conceptual fixed-wing aircraft design. While the program boasts a modular structure and a graphical user interface, the designer must manually select the analysis modules to arrive at the desired output. AAA possesses visualization tools, mainly in the form of tables and 2-D graphs. The geometry module, however, does allow for export of data to CAD programs that use AeroPack [68].

AAA is written in Pascal and FORTRAN, and uses a common database to store all design variables [69].

5.2 G.A.-CAD (Design, Analysis, and Research Corp.)

The General Aviation Computer Aided Design (G.A.-CAD) software was the successor to AAA; however, all of the added features have now been integrated directly into AAA. G.A.-CAD was designed to run on a personal computer (PC) and integrate a computer-aided drafting interface tailored for aircraft design [70]. Additionally, the software featured a choice of low or higher fidelity calculations in each of the modules.

5.3 RDS (Conceptual Research Corporation)

RDS is a PC-based sizing and analysis program, commercially available in both a student and a professional version. The student version performs first-order sizing, mission analysis, performance analysis, and cost analysis. Like G.A.-CAD, RDS has 2-D graphing capabilities, and integrates a specialized computer-aided drafting interface for 3-D visualization. Refer to Section 6.10 for information on RDS-Professional.

RDS is written in a combination of Turbo Pascal and Powerbasic. All variable information is saved and loaded from a single data file for each design [71].

RDS has several disadvantages. Very little in the analysis process is automated. The designer must manually enter all required data, in the appropriate order, to be able to perform specified calculations. While trade studies can be performed, they must be manually conducted (via user input variations).

5.4 ASCENT (Lockheed Martin)

The ASCENT program is formed from ASSET analysis routines and ACSYNT optimization modules [8]. ASCENT is not intended as a full-scale multidisciplinary optimization code. Instead, it is a high-fidelity tool for generating detail for more focussed trade studies of aircraft designs. Similar to RCD (refer to Section 6.13) in that various analysis disciplines can be selected, it also provides 3-D parametric geometry modelling capabilities for visualization and designer interaction and modification purposes.

5.4.1 ASSET (Lockheed Martin Skunk Works)

The main function of the Advanced System Synthesis and Evaluation Technique (ASSET) program is to size an aircraft to meet design requirements [8]. To do this, the software takes into account configuration design, propulsion, mass properties (structures, materials, and weights), aerodynamics, subsystems, cost, and program requirements. Detailed analysis is carried out to generate design parameters and parametric studies (using carpet plots).

5.4.2 ACSYNT (ACSYNT Institute)

The Aircraft Synthesis Tool (ACSYNT) is a versatile sizing, analysis, and optimization program. ACSYNT supports civil and military aircraft (including transport, fighter, bomber, and missile design) covering multidisciplinary aspects of geometry, trajectory (mission), takeoff and landing, aerodynamics, propulsion, stability, supersonic

characteristics, weights, and economic factors (more modules are planned, e.g. reliability, maintainability, and agility). The program features an interactive GUI, 2-D visualization tools (such as line graphs and carpet plots) for trade-off studies, and advanced 3-D visualization tools for parametric, feature-based surface modelling and shaded images.

The COPES program controls ACSYNT's modules and the optimization program CONMIN. ACSYNT's modules are written in FORTRAN, C, and C++. Current modules have been developed as object-oriented tools in C++ to alleviate the difficulties associated with software maintenance and integration with proprietary tools [72]. Data is stored in customized spreadsheets, and accessed by individual modules. CONMIN uses the constrained optimization method of steepest ascent. There are 916 design and constraint variables available for use, with typical studies ranging from 30-40 variables for practical considerations [73].

The main advantages of the ACSYNT program are the enforcement of software standards and the parametric feature modeller. Rigorous adherence to standards allow for program portability, continuity, maintenance, and ease of communication with other software. For example, the IGES standard is used to transfer configurations between ACSYNT and computer-aided drafting and manufacturing (CAD/CAM) programs. The parametric feature modeller allows the designer to build a design from stored components and optimize the design, or view the configuration resulting from an optimization and make modifications to the design (which can then be re-analyzed).

5.5 PASS (Stanford University)

The Program for Aircraft Synthesis Studies (PASS) has evolved into a modular analysis tool that uses a computational path generator, and expert advice and warning system for conceptual and preliminary design. The basic system structure is shown in Figure 5- 1 [10]. The executive routine provides a GUI in which the user can select from an aircraft design parameter database to view, modify, or compute variables. Two-variable trade studies can be performed and displayed in tabular or graphical form. The executive routine also provides the interface with the expert system and computational path generator.

Figure 5- 1: PASS System Structure

The PASS code is written in FORTRAN-77, and each routine is machine independent. Variables are stored in a database, and loaded into memory as necessary.

One unique feature of the PASS system is the quasi-procedural architecture for design variable computation. To compute a variable, the computational path generator links together only the appropriate procedural routines needed to compute the value of the variable. The system deals with all required inputs and outputs of each routine, and performs a heuristic search to execute an ordered path of routines for variable computation. Once a path has been generated, it can be reused. If the value of any variable along a computational path is later changed, the portion of the affected path is re-evaluated.

During computation, the expert system warns the user of problems in the design and offers advice on how to correct or improve the design. The system is based on sets of domain specific design rules, each of which are only included in the active knowledge set when required by the routines in use.

Refinements and additions planned for the system include using the computational path generator for system decomposition and parallel tasking, user preferences for multiple paths, and inclusion of a numerical optimization routine. Ideally, the expert system would be able to guide the optimizer to an improved design. More details are provided in [10].

5.6 CASDAT (Georgia Institute of Technology and Air Force Research Laboratory)

The Conceptual Aerospace Systems Design and Analysis Toolkit (CASDAT) is a collection of discipline-specific analysis tools for aircraft design, as shown in Figure 5- 2 [74]. The tools are predominantly UNIX-based, coded in FORTRAN, and have non-graphical interfaces. The CASDAT software provides a sophisticated GUI, with 2- and 3-D visualization capabilities (depending on the geometry model included). Built-in functions aid in the addition of modules to the framework.

Figure 5- 2: CASDAT System Structure

6. Conceptual Aircraft Design Optimization Software

A number of academic and commercial software packages have been developed for conceptual aircraft design optimization. The following subsections detail existing systems that may include proprietary algorithms, software, or commercial optimization frameworks.

6.1 OPDOT (NASA Langley)

The Optimum Preliminary Design of Transports (OPDOT) program was originally written as a preliminary optimization program, but it includes characteristics expected of a current conceptual design optimizer and was used in conceptual design studies [75,76,77]. OPDOT is designed to optimize an input design to an economic performance index while taking advantage of active controls and other technological developments.

OPDOT uses a direct sequential search simplex algorithm to optimize an input design for income required per flight for a fixed return on investment. Various constraints are integrated into an augmented objective function using penalty terms, depicted graphically in Figure 6- 1 [78]. Parametric studies in the form of 2-D graphs are also available.

Figure 6- 1: OPDOT System Structure

6.2 FLOPS (Kentron International, Inc.)

The Flight Optimization System (FLOPS) is an early multidisciplinary analysis and optimization program for conceptual aircraft configuration design and impact assessment of new technology. FLOPS features a modular, hybrid analysis structure (seen in Figure 6- 2 [79]), from which the designer can select the modules to be used in an analysis.

Figure 6- 2: FLOPS System Structure

While FLOPS is primarily an aircraft sizing and mission profile definition program, it can perform optimization on a limited number of design variables (9) and constraints (6). It uses a weighted objective function, and provides a choice of several optimization algorithms. Parametric studies in the form of 2-D contour plots are also available.

6.3 GASP + OPT (Lockheed-Georgia)

Another early attempt at conceptual aircraft optimization software combines an aircraft sizing program with an optimization algorithm. Figure 6- 3 [80] shows how the Generalized Aircraft Sizing Program (GASP) is integrated into a modular optimization framework that allows for changes in the algorithm type and sizing program adaptations. While sensitivity analyses can be obtained from the optimization, no visualization techniques are employed.

Figure 6- 3: GASP + OPT System Structure

The GRG method was chosen as the optimization algorithm, as described by Gabriele in [80], and incorporated into the optimization framework as the OPT program.

6.4 CASTOR (Loughborough University)

The Commuter Aircraft Synthesis and Trajectory Optimisation Routine (CASTOR) is an aircraft sizing and flight profile optimization program. CASTOR is limited to short-haul, twin-engine, propeller-driven commuter aircraft with conventional layouts [81].

CASTOR uses an unspecified form of single variable numerical optimization, written by the Royal Aircraft Establishment (RAE), which can handle a maximum of 25 design variables and 50 design constraints. Choice of minimum fuel, cost, or mass while

maintaining an optimum flight profile is provided for the measure of merit. Parametric studies are available in bar chart or graphical form.

6.5 ADAS (Delft University)

The Aircraft Design and Analysis System (ADAS) is an integrated program for analyzing and optimizing a given aircraft design. ADAS integrates the CAD program MEDUSA, as shown in Figure 6- 4 [modified from 5], to generate 2-and 3-D aircraft configurations. After an aircraft configuration is generated, the executive program can be run in parametric survey or optimization mode (utilizing an SQP algorithm), allowing the designer to perform both traditional and innovative design optimization.

Figure 6- 4: ADAS System Structure

Despite the fact that the system structure is integrated, the analysis codes (written in FORTRAN) are modular in nature and can be added or substituted in a tailored analysis run. However, the designer is required to have an extensive knowledge of programming to be able to compile and execute sequences of analyses. A number of analysis modules are included (e.g. weights, aerodynamics, load and balance analysis, direct operating cost, and range, field, and climb performance). The optimization mode is not modular, and can only work with a few design variables at a time.

2- and 3-D post-processing visualization techniques are included. Despite the extensive database, options to sort and compare different designs and to perform statistical analysis are not available [5].

6.6 TASOP (BAe Commercial Aircraft Ltd.)

The Transport Aircraft Synthesis and Optimisation Program (TASOP) is a hierarchical, modular aircraft synthesis and optimization program [65], similar in purpose to FLOPS and GASP. An outline of the system structure is shown in Figure 6- 5 [65]. TASOP is unique in that up to five missions can be defined, each with unique performance characteristics and payloads, for a single aircraft design. Thus, the aircraft is optimized for maximum capability and a range of typical operational environments.

Figure 6- 5: TASOP System Structure

TASOP is written in VAX FORTRAN, and an unspecified form of single variable numerical optimization (written by the RAE) is employed. 2-D parametric studies can be performed and output in graphical form.

While TASOP boasts a larger number of design variables (70) and constraints (50) than FLOPS, it is limited to fixed fuselage sizes and payloads for a given optimization. Expansion to include variation of fuselage size and payload while wing and/or engine size are held constant was planned as of 1990 [65].

6.7 CAPDA (Delft University)

The Computer-Aided Procedure for the Conceptual Design of Aircraft (CAPDA) is a modular sizing and aerodynamic performance optimization program. While the analysis and optimization modules are limited (as shown in Figure 6- 6), CAPDA does feature a graphical user interface, customized databases, and 2- and 3-D post-processing visualization techniques for configuration analysis [82]. A nice feature in post-processing is the use of bar charts and shaded models for relative configuration comparisons.

Figure 6- 6: CAPDA System Structure

As an optimization strategy, CAPDA uses a compromise between the multilevel optimization approach of Sobieszczanski-Sobieski [42] and a modified Hooke and Jeeves algorithm using external penalty functions.

The main disadvantage of CAPDA is the lack of flexibility in the initialization of a configuration. Mission parameters are not included, and the default geometry is based on an economy class transport jet from the statistical database. Many calculations are also based on the statistical information when detailed analysis modules are not included (e.g. weight). As a result, new technologies are difficult to implement and analyze.

6.8 PASS + NPSOL + GA (Stanford University)

The PASS system, as described in Section 5.5, has been effectively combined with NPSOL and a customized genetic algorithm [83]. NPSOL is a commercially available, gradient-based, constrained optimizer. The genetic algorithm was written to take advantage of the quasi-procedural program architecture used in PASS, without the customary employment of a binary design representation.

The advantage of this combination of programs is the ability to search the design space. The expert system in PASS can be used to generate multiple feasible starting points for the numerical optimization. The GUI and post-processing capabilities allow for easy comparison of optimized designs.

6.9 ACSYNT (ACSYNT Institute)

Refer to Section 5.4.2.

6.10 RDS-Professional (Conceptual Research Corporation)

RDS-Professional possesses all the features of the student version of RDS (see Section 5.3), as well as a parametric optimization module and higher fidelity analysis methods. The optimization module takes an existing design, sized and analyzed from the other modules, and optimizes it in the cycle shown in Figure 6- 7 [84].

Figure 6- 7: RDS-Professional Optimization System Structure

RDS-Professional employs a simple, gradient stepping optimization method. The quantity of design variables is limited to six, and the objective function is limited to a minimization of either weight or cost. Some performance constraints and variable bounds can be included [84].

6.11 AIDA (Delft University)

The Artificial Intelligence-supported Design of Aircraft (AIDA) program is a modular conceptual design tool that, unlike most of the existing software, can generate one or more concepts based on design specifications alone [85]. While AIDA employs minimal numerical optimization to aid in configuration determination, it does possess a number of notable features. Case-based reasoning is used to compare design specifications with existing designs for compatible start solutions. Rule-based reasoning (numerical relations expressing heuristics) is then used to evaluate and modify the start configurations to satisfy the constraints on the system. Figure 6- 8 [85] shows how AIDA is configured to take advantage of the case- and rule-based reasoning modules, as well as 3-D visualization through a geometrical module. The central user interface allows the designer to view different configurations and perform parametric studies (output in graphical format).

Figure 6- 8: AIDA System Structure

The case-based reasoning module is implemented in EADOCS, the functional module in QUAESTOR, and the geometrical module in Pro/Engineer. The central user interface was still under development as of 1998 [85], and research was unable to reveal documentation that is more recent.

6.12 BLISS (NASA Langley)

BLISS is a decomposition-based algorithm that uses GSE's or response surfaces for system optimization (refer to Section 3.5.3). The BLISS method has been coded for the conceptual design of a supersonic business transport, using a variety of modelling techniques and levels of subsystem fidelity.

The BLISS aircraft problem was written in both MATLAB and iSIGHT. Both optimization frameworks gave similar results for tests conducted with less than 30 design variables [52]. Subsystem analyses are modular, as shown in Figure 6- 9, and are performed in compatible codes (e.g. Excel spreadsheets, FORTRAN, ELAPS, and AWAVE). The subsystem modules can be replaced with higher fidelity analysis methods. Thus, the BLISS algorithm is applicable in any design level, from conceptual to detailed design [53].

Figure 6- 9: BLISS System Structure

The major advantage of the BLISS method is the ability to halt the optimization after each system optimization pass. This allows the designer to view the progress of the software and exercise judgement within the process.

6.13 RCD (Lockheed Martin Skunk Works)

The objective of the Rapid Conceptual Design (RCD) program was to develop a design tool to facilitate multidisciplinary design analysis and optimization at the conceptual level that would have the flexibility to address non-traditional design problems. RCD features a modular framework with automated scheduling and data transfer between modules [8].

The designer enters mission requirements and design constraints into the software. The designer then selects the analysis tools, associated fidelity level and data exchange to perform the optimization. The central controlling framework of the RCD program automates the scheduling of and data transfer between the analyses tools specified. All

system variables are available and monitored from the central framework location, while each analysis module operates on independent platforms. Figure 6- 10 (modified from [8]) shows a schematic of the RCD program. Because of the centralized, automated control program, updates or changes to a design are easily implemented throughout the entire system. Platform independent analysis modules allow simultaneous coordination on multiple computing hardware platforms. Analysis modularity allows for designer input and expansion of the software.

Figure 6- 10: RCD System Structure

Optimization is based on a weighted function of design variables. Plans to include a hybrid of genetic algorithms and gradient-based search methods for optimization have been defined.

7. Project Requirements

After examining the history of multidisciplinary design optimization software with respect to conceptual aircraft design, it is apparent that early MDO tools were limited in both scope and/or functionality. There exists a need for a new software tool for the conceptual design optimization of aircraft that addresses the issues that have arisen in the past with respect to automated MDO.

7.1 General MDO Software Requirements

In general, any MDO software should:

- be user-friendly.
- be of industrial strength. It should be able to handle a large number of design variables and constraints.
- be robust. Convergence should be attained under a wide range of conditions.
- be efficient, for reasonable levels of computing time.
- be reconfigurable. Nothing should prevent the designer from adding formalized routines, substituting judgement for a formal method, or overriding results. Provisions should also exist for a list of appropriate actions to be taken by the designer if deviating from standard procedure.
- have a logical, structured, modular design. This will aid in reconfiguration and enhance the program's extensibility.

- select what it needs to calculate based on the supplied information and objective function requirements.
- have a flexible method of data entry, with as little required data as possible.
- utilize relational databases. Possible development would include customized databases for existing aircraft, aircraft subsystems (e.g. engines, airfoils), and past projects.
- have the ability to accommodate a wide variety of informational sources such as estimates, statistics, references, experiments, look up tables, and other software.
- use many start points to explore the design space fully. Cold, warm, and hot starts should be supported.
- support interdigitation (a procedure by which a combination of different optimization algorithms are used to arrive at an optimal solution).
- inspire confidence in optimization results by allowing the designer to view the analysis and optimization process. The designer should also be able to interrupt the process at any point.
- have a fast, comprehensive, easy-to-interpret output format that fulfills previously determined visualization requirements.
- support interactive design changes to assess impact and perform re-optimization.
- provide full documentation and help files.
- support transition of generated data to preliminary design models and codes.

7.2 Software Requirements for Conceptual Aircraft Design

From the literature, the following salient features have been defined for conceptual aircraft design optimization software, compiled both from references included in the thesis and the author's own requirements. The software should:

- be able to handle a wide variety of aircraft types, both sub- and supersonic. This may include homebuilt props, single or twin props, pusher configurations, agricultural, regional turboprops, business jets, transport jets, military (transport, trainer, fighter, bomber, and patrol), unmanned vehicles, flying wings, and other unconventional configurations.
- use a generic geometric aircraft representation, implemented with a 3-D modeller.
- be able to generate conceptual designs from mission requirements, as well as optimize an existing design.
- support multiple flight conditions.
- have the ability to use alternate modules for different types of aircraft/analyses.

8. The Multidisciplinary Optimization of Conceptual Aircraft (MOCA) Project

In response to the defined need, a new research tool for the conceptual design optimization of aircraft is under development at Carleton University. With the aim of completion by several graduate students' effort over the next few years, the new MDO software development project for conceptual aircraft design is presented below. The area of research and development in this thesis is highlighted within the context of the project, and plans for future development are mapped out.

8.1 MOCA Software Structure

The MDO development project for conceptual aircraft design at Carleton University, dubbed the Multidisciplinary Optimization of Conceptual Aircraft (MOCA) program, will be formulated in a set of adaptable, reusable analysis blocks that can be dynamically configured to the problem at hand. The MOCA program will function entirely within an optimization framework, allowing access to all tools within the framework and operating on a predefined problem formulation. Figure 8- 1 shows the logo developed for the MOCA project.

Figure 8- 1: MOCA Project Logo

There are three assumptions made in the development of the MOCA program described herein.

- 1) The system is deterministic; that is, no uncertainty exists in the values of any design variables, in the accuracy of any of the design constraints, or within the formulation of the objective function.
- 2) No system decomposition will be used; the design variables and constraints may be separated into discipline specific modules, but numerical optimization will operate on the system as a whole.
- 3) The user of the software, designated hereafter as “the designer,” shall be given as much control over the selection of the optimization procedure as much as possible. As a result, the designer must have a thorough understanding of the optimization framework selected.

Figure 8- 2 illustrates the operation of MOCA, which evolved naturally from the requirements imposed on the software. The purpose of the software is either to allow a designer to generate one or several optimal conceptual aircraft designs from a set of mission requirements, or to optimize an existing design. Thus, the selection of a type of optimization to proceed with is the first option available to the designer when the program is run.

Figure 8- 2: MOCA System Flow Diagram

Figure 8- 3: Aircraft Type Selection and its Relation to the Module Categories

In the first case, the designer generates a near-optimal design from a set of mission requirements. Initially, an aircraft type must be selected. The aircraft type determines the modules available for use in the analysis of the system, as shown in Figure 8- 3, and links the appropriate database information. Mission performance parameters can be set and one or more mission profiles defined. The designer is then guided through a series of choices available to tailor the optimization procedure to the problem.

There are three possible levels of fidelity within each module category: Level 1 corresponds to a module containing empirical equations, Level 2 corresponds to an intermediate level of accuracy code (which may or may not be an individual program), and Level 3 corresponds to a high fidelity legacy or proprietary code integrated into the optimization framework. The default fidelity level for the system is set to Level 1. If an approximation is to be used for a specific module category or for the entire system, the next step allows the designer to set the type and extent of the approximation technique used. The default is “no approximation.”

The designer must select an objective function formulation, the design variables that are to be minimized, and the weighting of the variables (if a composite objective function is chosen). A choice of optimization algorithms is then available based on the objective function formulation. Once the designer has completed the optimization procedure selection, the program computes an initial design point, which may be infeasible. There are three types of initialization: cold, warm, and hot starts. A cold start uses a template design point solely based on aircraft type. A warm start searches through a historical

database for aircraft with similar mission requirements and configuration characteristics, and selects the closest match as an initial set of values. A hot start searches through a project database for similar optimizations performed previously, and uses the optimum design as a starting point for the new optimization. Hot starts are the most desirable, but since hot or, even, warm starts are not always possible, the program will select the best method and an initial design point within a given tolerance. The optimization procedure then executes on the initial design point to produce better solutions.

Throughout the optimization process, the designer is able to view the progress made by the optimizer through the visualization capabilities of the optimization framework. If the designer sees that the optimization is not progressing, the process can be interrupted at any point. Once a suitable solution has been generated, the data is made available through the post-processing capabilities of the optimization framework and output to a 3-D modelling program. Results can be modified or saved in the project database, and later re-optimized.

In the second case, MOCA allows a designer to optimize an existing aircraft design. Existing designs can be retrieved from the historical or project databases, or the data can be entered manually. Once the data on an existing aircraft is available, the optimization procedure selection is identical to that of the first case. The designer should estimate any missing information for the initial design point; individual analysis modules will ensure feasible values result in the optimal design. The post-processing and database options are the same as discussed previously.

8.2 Current Project Development

To implement the MOCA software project, there are three main areas of development for the thesis. The first area requires the development of the software structure within a selected optimization framework. Secondly, Level 1 modules for unmanned civilian aircraft will be compiled and tested to establish the viability of the structure. Module categories for this development will include Mission Definition, Initial Sizing, Geometric Configuration, Aerodynamics, Propulsion, and Performance Matching. Most of the work will focus on the Geometric Configuration module and a generic three-dimensional (3-D) parametric aircraft model for post-processing visualization purposes, which is the third area of major development. Implementation of a 3-D model will require integration of a computer-aided drafting package with the optimization framework.

8.2.1 Software Structure Development

The development of only Level 1 modules within the framework results in a simplified system structure. The modified system flow diagram proposed for initial software development is shown in Figure 8- 4. There are several differences between Figure 8- 4 and Figure 8- 2. The option for generating a number of conceptual aircraft designs from a set of mission requirements was eliminated to give the designer direct control (at this stage) over the aircraft configuration to be optimized. The option for manual selection of fidelity and approximations is not extant, but is still available for later development stages. In addition, the various aircraft type templates and the historical database will be left for future development (refer to Section 14).

The system structure is broken down into two current development stages. The first stage, shown in the grey path in Figure 8- 4, will establish the basic system structure and the ability to optimize an existing conceptual design from a set of mission requirements and geometric configuration parameters. The second stage, shown crosshatched in Figure 8- 4, will incorporate the ability to view the initial and final 3-D designs.

The driver behind allocation of tasks to each development stage was constrained by time and the decision to work in conjunction with the undergraduate fourth year aircraft design project in the Mechanical and Aerospace Engineering Department at Carleton University. In the fall term of the 2002/2003 academic year, the aircraft design project began the conceptual design process for an unmanned aerial vehicle (UAV). The fall term was devoted to generation of a number of conceptual aircraft designs to satisfy mission requirements provided by a customer. In the winter term, a single design was chosen and then developed in more detail. The fourth year project provides a unique opportunity to test MOCA through verification of results with conventional calculations. Additionally, it allows exploration of the area of the optimization of UAV's, where little research has been performed to date.

Figure 8- 4: Modified System Structure

8.2.2 Module Development

Due to time constraints, module development is limited to empirical equations of the kind typically used in an initial sizing routine. In this manner, the system structure can be developed and tested before complex analysis methods are integrated into the optimization framework.

8.2.3 Model Development

The geometric configuration, or OML, of an aircraft is extremely important in the conceptual design phase, as it determines much of the fundamental characteristics and performance of the aircraft. If the OML is effectively represented in the optimization process, a more optimal solution will result. While the OML can be represented as a set of design variables and constraints in the Geometric Configuration module, the designer will experience difficulty visualizing the configuration from a set of values without aid. An effective visual geometric representation is also required to allow the designer to assess the feasibility of the design rapidly.

As mentioned throughout the literature on the subject of conceptual aircraft MDO, no effective aircraft representation has been developed to date that can accommodate significant variations in configuration and highly unconventional designs. Therefore, a generic, 3-D, parametric model of an aircraft in terms of the design variables used in industry to express configuration layout must be developed that can be interfaced with a computer-aided drafting program. Due to the relative difficulty of such a task, this area of development will comprise a major portion of the author's work on the MOCA project.

9. Project Direction

To satisfy the requirements for the MOCA system structure as defined in Section 8, the development of a new MDO environment for conceptual aircraft design must first take the following steps:

- 1) Establish a mathematical model, parameterized in terms of the design variables, that is consistent with vehicle geometry and standard industry practices.
- 2) Identify information exchange requirements (input/output).
- 3) Select and perform a type of system decomposition.
- 4) Detail the optimization procedure, including objective function formulation, approximation techniques (if used), and optimization algorithm(s).

Since a number of mathematical programming languages, synthesis codes, and optimization frameworks are available commercially, the following tasks must be performed in conjunction with the above list:

- 1) Survey the existing software.
- 2) Select a framework suitable to the problem at hand, with the aim of minimizing the amount of coding and interface development.
- 3) Establish a data management system, if necessary.
- 4) Assemble the design-oriented analysis capability within the optimization framework.
- 5) Test the system analyses and optimization procedure capabilities.
- 6) Refine the system.

10. Software Selection

To minimize the amount of low-level programming required, the MOCA software tool for conceptual aircraft design at Carleton University will be created within an existing optimization framework. The framework will then utilize a mathematical programming language for complex analyses and a CAD package for 3-D parametric modelling.

The requirements for an optimization framework are defined for use in the MOCA project at Carleton University. A survey of commercial and non-commercial frameworks commonly used in the field of conceptual aircraft design is presented, and six are subsequently evaluated. An optimization framework is then selected for use in the MOCA project based on a trade study of qualitative requirements. In addition, the requirements and selection of an analysis environment and CAD program are also presented.

10.1 Selection of an Optimization Framework

An optimization framework is a problem solving environment that can support the implementation and execution of MDO applications and can provide a set of support services commonly required in applications of this type [86]. Essentially, optimization framework software provides all of the database, interface, communication, optimization, visualization, and debugging tools necessary to set up and evaluate an MDO problem.

Assuming a suitable optimization framework is selected, project development can focus on problem formulation, execution, and testing.

Recently, a number of companies have recognized the need for a software environment in which system modelling, analysis, and optimization can be carried out without a designer resorting to the use of low-level programming. Optimization frameworks have been developed to meet this need; providing software and system services to integrate computational tasks, a common working environment, and a full range of optimization tools [87]. With the selection of a suitable optimization framework, project development can focus on problem formulation, execution, and testing. The following two subsections detail the software and hardware requirements for an ideal framework.

10.1.1 Software Requirements for an Optimization Framework

Salas and Townsend [86] define four major categories of requirements for an MDO framework: architectural design, problem formulation, problem execution, and information access. The requirements from each category are subsequently adopted and augmented here.

Architectural Design

A framework should:

- provide a graphical user interface. The GUI should be user-friendly, efficient, and intuitive in nature.
- be designed using object-oriented principles. Object-oriented principles allow for features such as switching of analysis or optimization methods at run time and distributed computing.
- be extensible and should provide support for developing the interfaces required to integrate new processes into the system. The designer should be able to integrate new discipline codes, optimization algorithms, and other software tools of interest into the framework.
- not impose an unreasonable amount of overhead on the optimization process. Speed reduction may result from fine-tuning of the code, but should not result from software structure.
- be able to handle large problem sizes (on the order of 10^3 design variables and constraints).
- support collaborative design. To allow designers to work together, frameworks should allow simultaneous access to the problem data.
- be based on standards (e.g. STEP). Use of standards preserves investment and results in lower maintenance costs.

Problem Formulation

A framework should:

- allow the designer to configure complex branching and iterative MDO problem formulations easily without low-level programming (ideally using a visual programming interface).
- allow the designer to reconfigure existing MDO problem formulations easily. Fast reconfiguration supports the designer in exploring alternate views of the problem without reconstructing the problem each time.
- support the designer in incorporating legacy codes (written in a variety of languages) and proprietary codes (where the source code is unavailable) into the MDO problem formulation. The framework should be able to “wrap” such codes; that is, generate input files in the format required by such codes, invoke the codes, and extract the output of interest without any intervention by the designer.
- allow the designer to integrate discipline analyses with several optimization methods, including multilevel schemes involving sub-optimizations. The framework structure should not restrict any type of problem decomposition.
- provide facilities for debugging of multiple processes executing on computers across a network. The designer should be able to step through problem execution and monitor progress. Constructive feedback should be provided when problems are incorrectly constructed.

In addition, a framework should also provide a variety of automated approximation techniques and sensitivity analysis methods.

Problem Execution

A framework should:

- automate the execution of the processes and the movement of data. The framework should automate input file generation, problem execution, data extraction from output files, and the transfer of data between processes.
- be able to execute multiple processes in parallel. For computationally intensive problems, the designer should be able to identify and take advantage of coarse-grained parallelism.
- support execution distributed across a network of heterogeneous computers. Since higher-fidelity codes may exist on dedicated machines on a network, the framework should be able to access the machines across the network as necessary.
- support designer interaction (steering) during the design cycle. For example, the designer may want to substitute processes, disable execution of a process, switch optimization methods, adjust design variables, or activate/deactivate constraints.
- allow the designer to operate in a batch mode. This becomes important when using higher fidelity codes that have longer run times. The optimization process should be automatically scheduled and capable of running without designer intervention.

A framework should also provide a large number of optimization algorithms, support the addition of new algorithms, and be able to support interdigitation (the procedure by which various optimization algorithms are combined to execute numerical optimization of the system).

Information Access

A framework should provide:

- database management features. All information should be maintained in a central database, which the designer can access and manipulate.
- the capability to visualize intermediate and final optimization and analysis results. Visualization techniques are crucial for tracking the design history and interpreting results.
- a monitoring capability for viewing the status of an execution, including the system status.
- some mechanism for fault tolerance. A framework should be able to recover from a process failure with little or no loss of data.

The optimization framework chosen should also be within budgetary constraints of the project and capable of running on the available hardware.

10.1.2 Hardware Requirements for an Optimization Framework

The hardware available for this project is limited to the Mechanical and Aerospace Engineering department's graduate student computer lab. The following types of workstations were available at the time of software selection:

- Personal Computer (PC)
 - Windows NT Workstation 4.0, Service Pack 6a, installed on Pentium II (or higher) with a minimum of 128 MB of RAM

- UNIX
 - LINUX Redhat 7.1, minimum of 512 MB of RAM
 - SGI, IRIX 6.5.10, 128 MB of RAM

10.1.3 Optimization Framework Comparison

While separate numerical optimizers and analysis environments exist, as shown in Figure 10- 1, with a number of programs providing services to varying degrees along the spectrum, optimization frameworks must span the entire spectrum of services.

Figure 10- 1: Optimization Framework Spectrum

Ten software tools meet the fundamental definition of an optimization framework: DAKOTA, Epogy, FIPER, Genie, iSIGHT, LMS OPTIMUS, ModelCenter, OptdesX, SYSOPT, and VisualDOC. Due to a lack of current information, the Genie [88], OptdesX [89], and SYSOPT [45] frameworks will not be considered for the MOCA project. FIPER [90], though it promises to be a powerful environment, was not considered, as it has not yet been fully developed. The six remaining optimization frameworks are compared based on the requirements defined in Sections 10.1.1 and 10.1.2. Individual requirements in each requirement category are broken into feature-based and qualitative

components. The feature-based trade-off displays whether or not a framework possesses the specified feature, or if no information was available. The qualitative trade-off assigns numerical ratings to express how well a framework satisfies each requirement. It is important to note that the following comparisons are made on the basis of supplied literature, and a limited examination of demo, academic, or full versions of some of the software packages.

10.1.4 Feature-Based Trade-off Study

The results of the feature-based trade-off study are shown in Table 10- 1. A ‘Y’ indicates that the framework does possess the specified feature, an ‘N’ indicates that the framework does not possess the feature, and a ‘U’ indicates that it is unknown if the framework possesses the feature.

10.1.5 Qualitative Trade-off Study

The results of the qualitative trade-off study are shown in Table 10- 2. An integer rating is given between 0 (Poor) and 5 (Excellent) to express how well a framework satisfies each requirement. Cost is weighted double the value of other qualitative features since budgetary constraints are more significant.

10.1.6 Composite Trade-off Values

The totals for each trade-off study are combined in Table 10- 3. The composite results show that the iSIGHT optimization framework is the most comprehensive software tool and is best suited for the MOCA project.

Table 10- 1: Feature-Based Trade-off Study

	DAKOTA	Epogy	iSIGHT	LMS OPTIMUS	ModelCenter	VisualDOC
Architectural Design						
GUI	N	Y	Y	Y	Y	Y
Object-Oriented Design	Y	Y	Y	Y	Y	Y
Supports Collaborative Design	U	Y	U	U	Y	Y
Based on Standards	Y	Y	Y	U	Y	Y
Problem Formulation						
Accommodates Complex Problem Structures	Y	Y	Y	Y	Y	Y
Visual Programming Interface	N	Y	Y	Y	Y	Y
Incorporate Legacy and Proprietary Codes	Y	Y	Y	Y	Y	Y
Known Compatibility with Pro/Engineer	N	Y	Y	N	Y	N
Allows System Decomposition	Y	N	Y	U	N	N
Debugging Tools	U	N	Y	U	Y	Y
Provides Approximation Techniques	Y	Y	Y	Y	Y	Y
Provides Sensitivity Analysis Methods	Y	N	Y	Y	N	Y
Problem Execution						
Automated Processes and Data Transfer	Y	Y	Y	Y	Y	Y
Parallel Processing	Y	Y	Y	Y	Y	Y
Distributed Computing	Y	Y	Y	Y	Y	Y
Allows for Designer Interaction/Interruption	U	Y	Y	U	Y	N
Batch Mode Processing	U	Y	Y	U	Y	Y
Addition of Optimization Algorithms	Y	Y	Y	Y	Y	N
Interdigitation	U	Y	Y	U	N	N
Information Access						
Database Management Features	Y	Y	Y	Y	Y	Y
Monitoring of Problem Execution	N	Y	Y	Y	Y	Y
Fault Tolerance Mechanism	Y	Y	Y	Y	U	Y
Hardware						
Windows	N	Y	Y	Y	Y	Y
UNIX	Y	Y	Y	U	Y	Y
Total Y out of 24 Possible	14	21	23	15	20	19
Total out of 120 (Using Feature Weighting of 5)	70	105	115	75	100	95

Table 10- 2: Qualitative Trade-off Study

	DAKOTA	Epogy	iSIGHT	LMS OPTIMUS	ModelCenter	VisualDOC
Architectural Design						
Extensible	5	3	4	4	3	2
Rapid Analysis/Optimization	4	5	5	5	5	5
Large Problem Sizes	5	1	5	3	5	5
Problem Formulation						
Easy Reconfiguration	1	4	5	5	3	4
Problem Execution						
Multiple Optimization Algorithms	5	3	5	4	3	2
Information Access						
Visualization Techniques	1	3	5	5	4	2
Cost³						
Overall Package	10	3	6	7	4	9
Total out of 40	31	22	35	33	27	29

Table 10- 3: Composite Trade-off Results

	DAKOTA	Epogy	iSIGHT	LMS OPTIMUS	ModelCenter	VisualDOC
Feature-Based	70	105	115	75	100	95
Qualitative	31	22	35	33	27	29
Total out of 155	101	127	150	108	127	124

³ Cost information, available in Appendix A, was obtained through software vendor sales representatives. Contact information is available in Appendix B.

10.1.7 Conclusions

While the DAKOTA software is extremely attractive from a cost perspective, the lack of a GUI and the author's lack of experience with a UNIX environment make DAKOTA far less user-friendly than any of the other optimization frameworks.

Epogpy is a comprehensive package of optimization tools, but it is severely limited in the number of design variables a problem may contain (refer to Appendix A for more detail). Since the MOCA project may extend into preliminary design optimization, where thousands of variables may exist, Epogpy is inadequate.

iSIGHT is the most comprehensive of all optimization frameworks evaluated, possessing the most features and the best performance from a qualitative perspective. The cost of the iSIGHT software, as stated, is quite high, but Engineous, Inc. recognizes the benefits of academic use of their software and is willing to make alternative arrangements.

Little information was available on the exact capabilities of LMS OPTIMUS from the company website, customer support was unresponsive, and a demo version of the software could not be obtained. As a result, the OPTIMUS software performed poorly in the trade study.

ModelCenter is an excellent analysis integration framework, with limited optimization capability. While it would be possible to use ModelCenter for the MOCA project, it

would not be ideal. In addition, an evaluation copy of the software was provided and it was found that the software structure is not intuitive in nature, indicating a longer amount of time required to learn the software before problem implementation.

VisualDOC is highly intuitive in nature, albeit less flexible in overall structure. There are a number of limitations to the software, but none critical to the project at hand. After review of the demo version of the software, VisualDOC would be the second choice for use in the MOCA project despite faring slightly worse than Epogy and ModelCenter in the trade study.

10.1.8 Results

Based on the requirements outlined in Sections 10.1.1 and 10.1.2 for an optimization framework, the iSIGHT software was chosen for the MOCA project at Carleton University. Two floating licences for iSIGHT were obtained and the software installed.

10.2 Selection of an Analysis Environment

An analysis environment is a software program designed for mathematical computation.

10.2.1 Requirements for an Analysis Environment

Since the hardware requirements for any program used for the MOCA project remain the same as defined in Section 10.1.2, only the software requirements will be discussed here.

Essentially, an analysis environment should be:

- able to perform complex mathematical calculations.
- flexible enough to allow for user-defined functions.
- able to execute text-based script files to allow for integration with the optimization framework. While computations can be developed directly in the iSIGHT framework, the computational capability of the software is limited by the functions available in Tcl (Tool Command Language), the code in which iSIGHT is written. To take full advantage of additional functions and/or code that have already been developed, iSIGHT is designed to run simulation codes, or “simcodes.” Simcodes must be executable from the command prompt, and use ASCII (American Standard Code for Information Interchange) input and output files [91].
- readily available to the university at little or no additional cost. Ideally, the software licences would already have been purchased by the university.

10.2.2 Analysis Environment Selection

While a number of analysis environments exist (such as Matlab, Maple, Microsoft Excel, Mathcad, C++, FORTRAN, etc.), only Matlab fulfills all of the requirements defined in the previous section.

Matlab is an interactive software environment for mathematical computation and visualization. Computations are developed in a simple but powerful computing language and executed as scripts in ASCII text format, unlike Microsoft Excel or Mathcad. Matlab scripts are written in ASCII text and, as such, are amenable to integration within iSIGHT. Refer to Appendix C for detailed instructions on integrating Matlab as a simcode in iSIGHT. Matlab does not require detailed knowledge of computer programming syntax, and, as a result, scripts can be developed faster and easier as compared to languages like C++ or FORTRAN. However, the scripts must be run within the Matlab environment, and standalone executables cannot be generated without converting the scripts to a different programming language and compiling the code.

Carleton University has an existing contract for Matlab software licenses in the graduate student computer lab, and can be utilized at no cost to the project.

10.3 Selection of a CAD Program

A computer-aided drafting program allows for fast, automated generation and viewing of a design. A CAD program provides the designer with the ability to make changes to a design rapidly, and produce manufacturing information. The model can also be output in various forms for input into additional analysis software, such as FEA or CFD programs.

10.3.1 Requirements for a CAD Program

Since the hardware requirements for any program used for the MOCA project remain the same as defined in Section 10.1.2, only the software requirements will be discussed here.

The CAD program used for the MOCA project should:

- have 3-D solid modelling capability. 2-D packages are not as suitable for visualization purposes.
- employ parametric design for ease of design modification according to typical industry parameters.
- utilize assemblies to allow for integration of various model configurations.
- be able to execute text-based script files to allow for integration with the optimization framework.
- readily available to the university at little or no additional cost. Ideally, the software licences would already be purchased by the university.

10.3.2 CAD Program Selection

While a wide range of CAD programs exist (such as Pro/Engineer, AutoCAD, CATIA, IDEAS, and Solid Edge) with various capabilities, only Pro/Engineer fulfills all of the requirements defined in the previous section.

Pro/Engineer is a powerful 3-D parametric solid modelling program commonly used in industry for CAD/CAM applications. Assembly features are built into the basic modelling package (special add-on modules are not required). Pro/Engineer is able to import model data in the form of import blend (IBL) files, and execute automated model generation through ASCII text “trail” files. Trail files can also be parsed after execution for detailed analysis results. Refer to Appendix D for more information on integrating Pro/Engineer as a simcode into iSIGHT.

Carleton University has an existing contract for Pro/Engineer software licenses in the graduate student computer lab, and can be utilized at no cost to the project.

11. The MOCA Software Tool

The first two stages of development of the software tool for the MOCA project have established the basic system structure of the software tool within iSIGHT, the optimization framework chosen in Section 10.1, and the ability to optimize a conceptual aircraft design from a set of mission requirements and geometric configuration parameters. In addition to the design point visualization techniques included within iSIGHT, the initial and optimal design points can be exported to Pro/Engineer.

The calculations within the software code of MOCA are described in detail in a separate document [12], with in depth descriptions of the problem formulation, assumptions, and design constants. The general functionality of the software and each of the modules therein are summarized in the following subsections.

11.1 Optimization of an Existing Aircraft Design

The MOCA software tool is composed of three elements: a GUI to accept designer input, the analysis modules associated with the iSIGHT optimization framework, and the geometric configuration output to Pro/Engineer. Figure 11- 1 illustrates the system structure, including the analysis and optimization dataflow pathways. The input GUI appears as a separate interface and provides the information required to the “Set Initial Values” block in Figure 11- 1.

Figure 11- 1: The MOCA System Structure

The input GUI has been developed to lead the designer through a selection of aircraft types, and input of mission profile and geometric configuration parameter values. The information entered by the designer is then written dynamically to the optimization framework, where the designer can view the problem formulation, analysis modules, and tailor the optimization procedure.

Four sets of input from the designer are required before the optimization of an existing aircraft design can be performed. The initial input, shown in Figure 11- 2, consists of a selection between generating a new conceptual aircraft design and optimizing an existing design. The option for generating a new design (i.e. change in physical configuration) is not available at this stage in the development of the MOCA project. Secondly, the designer must then select a type of aircraft to optimize, in a similar window display. While the viewable aircraft types match the selections available in Figure 8- 3, only the option for a small project, subsonic, civilian UAV (computed in imperial units) is currently active.

Figure 11- 2: The Initial MOCA GUI

The third input requirement is for mission profile data. At this stage, only a single, flexible mission profile defined for unmanned civilian purposes (i.e. no combat segment)

is offered. The profile can be tailored to suit a range of missions simply by including or excluding data on a particular mission segment. The only mandatory elements include warm up, takeoff and cruise conditions; Segment 0-1 and Segment 2-3, respectively, illustrated in Figure 11- 3. The mission profile input is required to perform initial sizing of the aircraft based on the method described by Raymer [92], and performance matching constraints based on the methods of Loftin [93] and Roskam [94]. Refer to Section 11.2 for more information on the analysis modules employed.

Figure 11- 3: The Mission Profile GUI, Including the Help Dialogue

The last input GUI, shown in Figure 11- 4, is comprised of geometric configuration options and parameters. The geometric configuration options allow the designer to specify physical characteristics of the aircraft that are not defined with numeric values, such as material type. Table 11- 1 lists all options associated with each characteristic that are presently available; additional options and new characteristics can easily be added in the future (refer to [12]). Independent selection of the various characteristics allows for a wide variety of overall aircraft configurations, including a number of highly unconventional designs. Figure 11- 5 illustrates a sampling of the possible configurations, not including engine placement.

Configuration Options

Select a material type for the aircraft: Metal/Wood

Select fuselage symmetry about wing center: Symmetric

Select a fuselage cross section shape: Rectangular

Select wing placement: Low Wing

Select a canard type: No Canard

Select a canard placement: Low Canard

Numeric Parameters

Number of Fuselages	0
Maximum Fuselage Diameter	0 ft
Fuselage Aspect Ratio (W/H)	0
Wing Span	0 ft
Wing Aspect Ratio	0
Wing Root Chord	0 ft
Wing Tip Chord, Starboard	0 ft
Wing Tip Chord, Port	0 ft
Wing Oblique Angle	0 deg
Wing Dihedral 1	0 deg
Wing Dihedral 2	0 deg
Wing Dihedral 3	0 deg
Wing Dihedral 4	0 deg
Wing Dihedral 5	0 deg

Please enter an initial estimate for the gross takeoff weight of the aircraft: 0 lbs

< Back Finish > Cancel

Figure 11- 4: The Geometric Configuration GUI

Table 11- 1: Geometric Configuration Options

Variable	Value
Aircraft Material Type	Metal/Wood
	Composite
Fuselage Symmetry	Symmetric
	Asymmetric
Fuselage Cross-Section Shape	Rectangular
	Conic
Wing Placement	Low Wing
	Mid-Wing
	High Wing
Canard Type	No Canard
	Non-Lifting Canard
	Lifting Canard
Canard Placement	Low Canard
	Mid-Canard
	High Canard
Horizontal Tail Type	None
	Conventional
	Cruciform
	T-Tail
	Single or Twin Boom Mounted
	H-Shape
Vertical Tail Type	None
	Conventional
	V-Shape
	Y-Shape
	Single or Twin Boom Mounted
	Twin Boom/Fuselage Mounted
	Inverted V
Engine Type	H-Shape
	Propeller
Engine Placement	Jet
	Fuselage, Internal or External
	Tail Combination
	Wing Combination
	Multiple Fuselage Combination
Engine Direction	Tractor
	Pusher
Landing Gear/Wing Configuration Type	Retractable, Cantilevered Wing
	Fixed, Cantilevered Wing
	Fixed, Strut-Braced Wing

Figure 11- 5: Sample Aircraft Configurations

Engine placement is highly dependent upon the number of engines, number of fuselages, fuselage symmetry, and horizontal and vertical tail types, as well as engine direction in the case of propeller-driven aircraft. Boom placement is quite similar. The figures in Appendix E depict the some of the most common engine placements for propeller and jet engines, as well as boom placements; those noted in white boxes have been implemented

within the MOCA software tool, while those in grey boxes have not. Nor are these figures exhaustive or detailed about the possibilities for locations, rather, they are meant as a starting point for consideration of the conceptual design and subject to further refinement.

From Table 11- 2, the numeric parameters are self-explanatory with the exception of lifting surface parameters followed by a number (e.g. Wing Dihedral 1). The wing and canard surfaces are broken into a maximum of six sections, symmetric or asymmetric, with two breakpoints each per side of the centerline as shown in Figure 11- 6. Each breakpoint is defined as a positive distance from the centerline and numbered from starboard to port. Each section of the lifting surface is then numbered from starboard to port and all parameters associated with the section are given the same number. Hence, Leading Edge Sweep 1, Dihedral 1, and Taper Ratio 1 all apply to section 1. Twist 1 applies to the initial chord line of section 1 (i.e. the tip twist), Twist 2 applies to the chord line at Breakpoint 1, and so on (linear interpolation is used between breakpoints to determine the twist at individual wing stations). If only one breakpoint per side is specified, Breakpoints 2 and 3 degenerate to the root chord position. If no breakpoints are specified, Breakpoints 2 and 3 degenerate to the root chord position and Breakpoints 1 and 4 degenerate to the tip chord positions. Data for all six sections must still be entered using the appropriate degenerate locations⁴.

⁴ For example, if only one breakpoint is set at a distance of 10 ft from the centerline, Breakpoints 1 and 4 equal 10, while Breakpoints 2 and 3 equal zero. Leading Edge Sweeps and Dihedrals 2 and 3 are also set to zero, while Taper Ratios 2 and 3 are set to one to maintain continuity of the planform. All values for points 1 and 4 are given the actual value for the aircraft to be optimized. It is important to remember that the taper ratio is for the specified section (not tip/root).

Table 11- 2: Geometric Configuration Parameters

Variable	Allowable Values
Fuselage Number	1,2,3,4
Maximum Fuselage Diameter	>0
Fuselage Aspect Ratio	>0
Wing Span	>0
Wing Aspect Ratio	>0
Wing Root Chord and Tip Chord (Starboard and Port)	>0
Wing Oblique Angle (Entire Wing)	between -90 and +90 degrees
Wing Dihedral 1-6	between -90 and +90 degrees
Wing Incidence	between -90 and +90 degrees
Wing Twist 1-6	between -90 and +90 degrees
Wing Leading Edge Sweep 1-6	between -90 and +90 degrees
Wing Taper Ratio 1-6	between 0 and 1
Number of Wing Breakpoints	0,1,2
Wing Breakpoint Y-Distance 1-4	>=0
Percent of Lift Carried By Wing	between 50 and 100%
Static Margin	between 5 and 20%
Canard Span	>0
Canard Aspect Ratio	>0
Canard Root Chord and Tip Chord (Starboard and Port)	>0
Canard Oblique Angle (Entire Canard)	between -90 and +90 degrees
Canard Dihedral 1-6	between -90 and +90 degrees
Canard Incidence	between -90 and +90 degrees
Canard Twist 1-6	between -90 and +90 degrees
Canard Leading Edge Sweep 1-6	between -90 and +90 degrees
Canard Taper Ratio 1-6	between 0 and 1
Number of Canard Breakpoints	0,1,2
Canard Breakpoint Y-Distance 1-6	>=0
Horizontal Tail Span	>0
Horizontal Tail Aspect Ratio	>0
Horizontal Tail Oblique Angle	between -90 and +90 degrees
Horizontal Tail Dihedral 1-2	between -90 and +90 degrees
Horizontal Tail Leading Edge Sweep 1-2	between -90 and +90 degrees
Horizontal Tail Taper Ratio 1-2	between 0 and 1
Horizontal Tail Incidence	between -90 and +90 degrees
Horizontal Tail Twist 1-2	between -90 and +90 degrees
Vertical Tail 1,2,3 Rotation Angle	between 0 and 360 degrees
Vertical Tail 1,2,3 Span	>0
Vertical Tail 1,2,3 Aspect Ratio	>0
Vertical Tail 1,2,3 Leading Edge Sweep	between -90 and +90 degrees
Vertical Tail 1,2,3 Taper Ratio	between 0 and 1
Vertical Tail 1,2,3 Incidence	between -90 and +90 degrees
Vertical Tail 1,2,3 Twist	between -90 and +90 degrees
Number of Booms	0,1,2
Number of Engines	1,2,3,4
Gross Takeoff Weight Estimate	>0

It is also important to note that a breakpoint can be set for a change of value of any of the previously mentioned variables or combination thereof, as well as chord length (to obtain instantaneous changes in chord, common to delta style wings).

Figure 11- 6: Lifting Surface Planform

The horizontal tail is assumed to have no breakpoints and can be either symmetric or asymmetric. There are only two sections to the lifting surface and the same numbering scheme applies. The vertical tail is assumed to have no breakpoints and only one section. However, depending on the choice of vertical tail type, there may be up to three vertical tail sections – each with its own parameter values. Typically, the vertical tails will be identical except for the angle at which they are canted. The cant (or rotation) angle is

defined as a positive angle measured clockwise in the front view of the aircraft from the starboard horizontal plane to the zero lift line of the vertical tail, assuming zero twist and camber.

11.2 Analysis Modules

The analysis modules included thus far in the MOCA program include Propulsion, Geometric Configuration, Aerodynamics, Initial Sizing, and Performance Matching. Each analysis module is either a task block in iSIGHT or a Matlab M-File, both of which allow for easy access and modification of the computations. The following subsections provide a brief summary of the problem formulation within each of the analysis modules; detailed information on the calculations is provided in reference [12], including all assumptions, constants, and variables used. All modules are written as tasks within iSIGHT unless otherwise noted.

11.2.1 Propulsion

The Propulsion module is simply a conversion between the equivalent thrust to weight ratio, power loading, and power to weight ratio. All calculations for jet aircraft are performed using the thrust to weight ratio, while propeller aircraft use power loading and power to weight ratio where appropriate, except in the Performance Matching module. The equivalent thrust to weight ratio is used to evaluate the performance constraints in the same units as jet aircraft for ease of comparison (refer to Section 11.2.5).

11.2.2 Geometric Configuration

The Geometric Configuration module is a Matlab M-File called as a simcode in iSIGHT. The role of the module is to generate the physical aircraft representation in a file format that Pro/Engineer can read and use to develop the 3-D solid model of the aircraft. The calculations performed within the module enforce realistic physical consistency within the representation wherever possible, while attempting to accommodate as wide a range of fixed-wing aircraft configurations in as simple a manner as possible. The module is divided into eight defining components of the aircraft configuration: the fuselage, wing, canard, wing strut, boom(s), horizontal tail(s), vertical tail(s), and representative engine(s). All components except for the fuselage and wing are optional.

Figure 11- 7: Fuselage Sectioning

The fuselage can be either rectangular or conic in cross section shape, and is broken down lengthwise into three main components as seen in Figure 11- 7. The nose cone, constant cross section body, and tail cone are controlled by fineness ratios (length/diameter), relating the overall length of the fuselage to the maximum diameter of the fuselage specified by the designer. The nose cone is hemi-ellipsoid in shape to match the slope of the surface where it meets the constant cross section body. The tail cone

matches the maximum height of the fuselage while tapering underneath to a small hemispherical end at a set angle (the angle is highly dependent on the fineness ratio of the tail cone and so is not yet directly alterable by the designer).

The wing is a horizontal lifting surface whose planform shape is defined in Figure 11- 6. Airfoil coordinates as a function of chord length, read from a text file, describe the upper and lower lifting surface. At this time, only 4-digit series NACA airfoils are provided. Additional airfoils may be included if the coordinates are written in text files in the same format as those already included (refer to [12] for more details). While the option is not available to the designer through the MOCA input GUI, the wing representation is capable of handling multiple airfoils with linear interpolation between breakpoints. Wing tip additions such as winglets or endplates have not yet been incorporated into the Geometric Configuration module, but are planned (refer to Section 14). Variable sweep cannot be accommodated due to the static nature of the solid model produced.

The wing and canard are modelled in exactly the same fashion. This was done deliberately to accommodate many complex lifting surface configurations, such as a non-lifting canard, lifting canard, a joined wing, and biplanes. In the non-lifting or control canard case, the canard is placed using a volume coefficient, while in the lifting canard case, the wing and canard placements are controlled by the ratio of the amount of lift produced by each surface and the requisite static margin utilizing a simplified longitudinal static stability analysis.

Two optional additions to the wing and fuselage include struts and booms. A simple, airfoil shaped wing strut was included as a possible configuration option in either high or low wing cases to represent a typical homebuilt style. Tail and twin booms are common UAV styles, and were included with the option of fuselage or boom-mounted horizontal and vertical tails.

The horizontal and vertical tails possess simplified planforms of the wing configuration. The horizontal tail is broken into two sections, one on either side of a fuselage or boom centreline mount, and the vertical tail is a single section starting at the centreline of a boom or fuselage mount. The tail surfaces are assumed to use symmetric airfoils (i.e. no camber) and are sized using volume coefficients. The possible horizontal and vertical tail combinations are listed in Table 11- 1.

Representative engines have been included in the Geometric Configuration module to give the designer a better idea of the feasibility of the aircraft configuration at the conceptual stage. Sizing of the engine(s) are based on the required takeoff horsepower for propeller engines and takeoff thrust for jet engines. Correlations, available in reference [12], were developed from engine performance and physical data in [95], [96] and [97]. As a result, the designer is able to assess if an engine roughly in the power region required will physically fit within the fuselage (if placed there), and how the engine will look. If an engine is not placed within the fuselage, a representative nacelle is developed around the required engine size and located on the aircraft (refer to Appendix E for possible locations). Detailed development is required to bring the engine(s) sizing and

placement to a level of sophistication on par with the remainder of the Geometric Configuration module (see Section 14).

11.2.3 Aerodynamics

The Aerodynamics module calculates the Oswald efficiency factor for the wing [98], and then estimates the parasite drag of the aircraft as per Roskam [94]. The lift to drag ratio is then found for subsonic aircraft, neglecting drag due to compressibility effects.

11.2.4 Initial Sizing

The calculations within the Initial Sizing module are dependent upon the mission profile established in Section 11.1. The sizing method is taken directly from Raymer [92] for both propeller and jet engine aircraft⁵. The weight estimate input from the designer is used as the starting point for iteration through the initial sizing routine. The module terminates when the gross takeoff weight is within 0.01 lbs between iterations, due to misleading optimization results when convergence to a large number of significant figures is not obtained. It should be noted that the 0.01 lbs tolerance is not an indication of the accuracy of the gross takeoff weight produced by the Initial Sizing module.

11.2.5 Performance Matching

The Performance Matching module is a Matlab M-File called by iSIGHT as a simcode. Essentially, the method is the same as that used by Loftin [93] and Roskam [94] but adapted to suit an automated environment.

⁵ While the MOCA program is currently configured for unmanned aircraft, the initial sizing routine can also handle manned aircraft. Crew and passenger weights are simply set to zero at the start.

Performance objectives specified by the designer in the MOCA input GUI are translated into constraints on the thrust to weight ratio (T/W) and wing loading (W/S) of the aircraft. The allowable range of values of T/W and W/S in which the performance requirements are met is considered the feasible design space. The current constraints can include sea level stall speed, takeoff distance (assuming FAR 23 requirements for Civilian Small Project Aircraft, propeller and jet under 12,500 lbs), sea level climb rate, cruise speed, maneuver, and landing distance (assuming landing immediately after takeoff and FAR 23 power off approach requirements for General Aviation Aircraft). Since some of these can be met in either flaps up or flaps down condition five typical high lift device options with predetermined performance, in increasing lift and complexity [99], are available for use with any given aircraft configuration. Using a DOE study, iSIGHT evaluates each high lift device setting to see if any of the performance constraints are violated, and selects the least complex device that satisfies all constraints. If the most complex device cannot satisfy the constraints, then a constraint violation is recorded and passed back to the optimization in iSIGHT.

The constraint violation is a linear combination of the shortest distance from the design point to each constraint line when plotted on a W/S versus T/W graph. The effect of the composite constraint violation can be seen in Figure 11- 8 when plotted against W/S and T/W . The flat region at the bottom of the graph where the constraint violation is zero is the feasible design space. Each distinct change in slope in the surface is the effect of an additional constraint being violated. By comparing Figure 11- 8 and Figure 11- 9, the

large increase in feasible design space can be seen due to the increased performance of the high lift devices. The larger the design space, the more likely the design point is feasible.

Figure 11- 8: Performance Matching Constraint Violation, No Flaps

**Figure 11- 9: Performance Matching Constraint Violation,
Slats and Triple Slotted Flaps**

Roskam [94] plots the constraints in two dimensions and advises the designer to select a design point that minimizes T/W and maximizes W/S while remaining inside the feasible region, and, as a result, will find the design with the lowest weight and cost. By minimizing gross takeoff weight in iSIGHT (see next subsection) while constraining the composite constraint violation value to zero, the process works in reverse to optimize the design point, and consequently, minimizes T/W and maximizes W/S .

11.3 Optimization Problem Formulation

Presently, the objective function for the optimization problem is a simple minimization of gross takeoff weight. The objective function can be modified at any time to include additional design variables, as well as weighting factors. The independent design variables include thrust to weight ratio, wing span and aspect ratio. The remainder of the design variables are dependent or become dependent once the optimization begins (for example, the wing root chord is expressed as a constant percentage of the wing span and the chord value is updated as the wing span varies).

11.4 Optimization Algorithms and Approximation Techniques

There are thirteen individual optimization algorithms available within iSIGHT. Of the thirteen, four are not appropriate for the given problem formulation. The Multi-Island Genetic and Adaptive Simulated Annealing algorithms are stochastic methods utilizing a more random sampling of the design space; the current problem formulation requires an iterative method that closes in on the design optimum as the gross takeoff weight from one step is used in the next step, before the initial sizing routine updates the value. The Method of Feasible Directions – CONMIN algorithm does not allow equality constraints, and the Successive Approximation Method aborts early in the optimization (refer to Section 12.1.5 for more information on the optimization algorithms tested with the MOCA problem formulation). The remaining nine algorithms may be used within MOCA:

- Sequential Quadratic Programming - NLPQL
- Mixed Integer Optimization - MOST
- Generalized Reduced Gradient - LSGRG2
- Sequential Quadratic Programming - DONLP
- Exterior Penalty Method
- Modified Method of feasible Directions (MMFD)
- Sequential Linear Programming (SLP)
- Hooke-Jeeves Direct Search Method
- Directed Heuristic Search

No approximation techniques are currently in use; approximations can be developed subsequently in iSIGHT if desired.

11.5 Optimal Design Visualization

Two types of visualization techniques are employed in the MOCA program. The non-artifact, non-topographical based tables and 2-D graphs available within iSIGHT are customizable, displaying input or output variables selected by the designer both during and after a design optimization run. The artifact-based Pro/Engineer aircraft representation is designed to show the optimal design point reached after an optimization run. The information from the starting design point is stored and can be retrieved later for visual comparison of the initial and final designs.

11.6 MOCA Program Feature Summary

The MOCA software tool is a multidisciplinary design optimization program for conceptual aircraft design that is notable for the following features:

- It is currently able to perform optimization of unmanned air vehicles, and provisions are in place to include multiple types of manned aircraft in the future.
- It utilizes a flexible mission profile.
- The designer can view the reconfigurable analysis modules (no hidden, unalterable code).
- The objective function can be redefined by the designer to reflect almost any objective, including composite functions.
- It provides easy linking to other analysis codes through simcodes in iSIGHT.
- All of the benefits of the iSIGHT optimization framework are available for the designer to use (refer to Section 10.1).
- It employs a comprehensive geometric representation of the aircraft for accurate, direct visualization before and after the optimization.
- The automatic generation of a functional 3-D solid model shortens the conceptual design process and provides a foundation for preliminary design work. The Pro/Engineer model can be used as the basis for further solid modelling and file generation for machining of wind tunnel models, rapid prototyping models, finite element analysis, computational fluid dynamic codes, and more.

12. Testing of the MOCA Software Tool

Several levels of testing of the MOCA software tool were performed to ensure the validity and applicability of the results. After development of the initial sizing module in the MOCA software, it was deemed necessary to verify the results of the analysis methods used therein with a sample conceptual aircraft design. By comparing the results to similar hand calculations, the calculations in MOCA can be checked for accuracy. Once the analyses were proven acceptable, optimizations of two dissimilar test cases were performed.

The testing of the MOCA software tool was completed on a 1.0 GHz Pentium III laptop PC with 256 Mb of RAM running Windows XP Professional. The options for distributed computing and parallel tasking were available, but not utilised to show the typical upper limit on the amount of time required to perform the analyses and optimizations.

12.1 Test Case 1

The first test case was selected from the 2002/2003 undergraduate fourth year aircraft design project, which completed the conceptual and preliminary phases of the design of a UAV for geophysical survey work. Section 12.1.1 details the mission requirements used for the project, which are adopted here for the sample design. Section 12.1.2 includes all of the geometric parameters of the aircraft as of the end of the academic year.

The choice of this conceptual design for comparison of the initial sizing analyses provides several advantages over an existing UAV design. Hand calculations are readily available for comparison, the aircraft design can be compared at the appropriate design phase (as opposed to long after design was completed), and extensive familiarity with the project (the author participated as a lead engineer for the Design Integration Group). Optimization of this test case while in the early design phases also provides important feedback to the project to allow the students to develop a more optimal design.

12.1.1 Sample Mission Requirements

In conjunction with Sander Geophysics Ltd. of Ottawa, ON, the 2002/2003 aircraft design project developed a set of mission requirements for a new long-endurance unmanned aerial vehicle for geophysical survey work (GeoSurv UAV). The mission requirements are as follows:

Endurance:	14 hours at 100kts, at ISA conditions
Cruise Speed:	60-100 kts
Combined Payload Weight:	1000 lbs
Combined Payload Volume:	22 ft ³
Sea Level Stall Speed:	45 kts
Cruise Altitude:	150 ft above ground level
Dash Speed:	150 kts at 2500 ft above sea level
Sea Level Rate of Climb:	1000 ft/min
Takeoff and Landing Field Length:	2000 ft, ISA

Service Ceiling:

20,000 ft above sea level

12.1.2 Geometric Configuration

The GeoSurv UAV is a low-speed, low-flying tractor propeller aircraft. The fuselage is unpressurized and composed of aluminum. It has a conventional horizontal and vertical tail configuration with a high, strut-braced wing and fixed landing gear. The GeoSurv UAV is also equipped with a tail boom to house sensors used for geophysical surveying. Table 12- 1 summarizes the geometric configuration options input into the MOCA software tool. Table 12- 2 provides the numeric values of the geometric configuration parameters for the GeoSurv UAV at the end of the 2002/2003 preliminary design phase.

Table 12- 1: GeoSurv UAV Geometric Configuration Options

Variable	Value
Aircraft Material Type	Metal
Fuselage Symmetry	Symmetric
Fuselage Cross-Section Shape	Rectangular
Wing Placement	High Wing
Canard Type	No Canard
Horizontal Tail Type	Conventional
Vertical Tail Type	Conventional
Engine Type	Propeller
Engine Placement	Fuselage, Internal
Engine Direction	Tractor
Landing Gear/Wing Configuration Type	Fixed, Strut-Braced Wing

Table 12- 2: GeoSurv UAV Geometric Configuration Parameter Values

Variable	Value
Fuselage Number	1
Maximum Fuselage Diameter	3.5
Fuselage Aspect Ratio	1.0
Wing Span	48.00
Wing Aspect Ratio	9.18
Wing Root Chord	6.15
Wing Tip Chord (Starboard and Port)	4.31
Wing Oblique Angle	0
Wing Dihedral 1-6	0
Wing Incidence	1.0
Wing Twist 1,2,5,6	-2.5
Wing Twist 3,4	1.0
Wing Leading Edge Sweep 1-6	0
Wing Taper Ratio 1,6	1.00
Wing Taper Ratio 2,5	0.878
Wing Taper Ratio 3,4	0.797
Number of Wing Breakpoints	1
Wing Breakpoint Y-Distance 1,4	24.00
Wing Breakpoint Y-Distance 2,3	16.25
Horizontal Tail Span	14.74
Horizontal Tail Aspect Ratio	5
Horizontal Tail Oblique Angle	0
Horizontal Tail Dihedral 1-2	0
Horizontal Tail Leading Edge Sweep 1-2	18.8
Horizontal Tail Taper Ratio 1-2	0.4
Horizontal Tail Incidence	-0.6
Horizontal Tail Twist 1-2	-0.6
Vertical Tail 1 Rotation Angle	90
Vertical Tail 1 Span	6.33
Vertical Tail 1 Aspect Ratio	1.5
Vertical Tail 1 Leading Edge Sweep	18.4
Vertical Tail 1 Taper Ratio	0.4
Vertical Tail 1 Incidence, Twist	0
Number of Booms	1
Number of Engines	1
Gross Takeoff Weight Estimate	3000

12.1.3 Trial 1

The first trial performed by the MOCA software tool was used to verify the initial sizing calculations before the geometric configuration module was completed. Results are compared with two sets of hand calculations to verify the analysis methods used. All three methods use the sample mission requirements to begin calculations. While the analysis methods within MOCA are presented in detail in [12], the differences between MOCA and the other two methods are outlined below. The calculations provided by the GeoSurv project team are contained in [100].

Propulsion characteristics are generated from historical data, except in the hand calculations where performance matching is used to select a design point for power loading and wing loading. The geometric configuration is determined by designer input, which is, in this case, the GeoSurv project team. The hand calculations are independent of these parameters. Aerodynamic characteristics are initially determined from historical data in MOCA, and from a drag polar in the other two methods and later stages of MOCA. Each method performs the initial sizing routine in the same fashion, with the exception of the source of the empty weight fraction. MOCA uses a curve fit of historical data to predict the empty weight fraction, while the GeoSurv project team simply selected a conservative value based on a survey of several UAV's with similar mission requirements. The hand calculations use an average, conservative value from a database of eleven UAV's. All three methods produce gross takeoff weight as the main output parameter. Also included are values for empty weight and fuel weight.

Values calculated from each method of analysis are tabulated according to module organization within MOCA and presented in Table 12- 3. The values are quite similar, with minor variations due to the specific assumptions and analysis methods used, confirming that the calculations in MOCA are producing correct results.

Table 12- 3: Initial Sizing Comparison

		Type	MOCA	GeoSurv Calculations	Hand Calculations	Units
Analysis Modules	Set Initial Values	Crew Weight	Input	0		lbs
		Fixed Payload Weight	Input	1000		lbs
		Dropped Payload Weight	Input	0		lbs
		Total Payload Weight	Input	1000	1000	lbs
		Engine Type	Input	Propeller	Propeller	
		Material Type	Input	Aluminum	Aluminum	N/A
		Landing Gear Type	Input	Fixed	Fixed	
		V _{max}	Input	100	100	kts
		Cruise Altitude	Input	150	150	ft
		M _{engine}	Output	0.15124		
	Propulsion	Power Loading	Output	14.5		lbs/hp
		Thrust To Weight Ratio	Output	0.180	0.202	
		C _{thp} Cruise	Output	0.4	0.4	lbs/hr/hp
		C _{thp} Loiter	Output	0.4		lbs/hr/hp
	Geometric Configuration	Wing Span	Input	48	48	ft
		Aspect Ratio	Input	9.18	9.18	
		Taper Ratio	Input	0.7	0.7	
		# of Wing Breakpoints	Input	1	1	
	Aerodynamics	Parasite Drag	Input	0.0207	0.0207	
		Lift To Drag Ratio	Output	14.5	14	
		Lift To Drag Ratio Max	Output	16.7	17.0	
		Propeller Efficiency	Input	0.8	0.8	
	Initial Sizing	Takeoff Fuel Fraction	Input	0.970	0.970	
		Descent Fuel Fraction	Input	0.990		0.993
		Landing Fuel Fraction	Input	0.995	0.995	0.993
		Range	Input	1400	1400	nm
		Endurance	Input	0	0	hr
		Percent Reserve Fuel	Input	5	5	
		Percent Trapped Fuel	Input	1	1	
		Takeoff Wing Loading	Output	13.32	13.5	12.6
		Empty Weight Fraction	Output	0.518	0.51	0.514
		Empty Weight	Output	1734	1733	1683
		Fuel Weight	Output	611	665	592
		Gross Takeoff Weight	Output	3345	3397	3275

Legend: 1 Constants
2 Constants Chosen Based on Historical Data
3 Calculated From Empirical Equations
4 Calculated Based on Historical Data

12.1.4 GeoSurv UAV Drag Estimation

Before optimizing the GeoSurv UAV conceptual design, the drag polar coefficients in the Aerodynamics module were tailored to be more representative of the actual aircraft configuration. The drag polar, given by Equation 12-1, is composed of the parasite drag coefficient, C_{D0} , and the induced drag, which is a function of the lift coefficient (C_L) of the aircraft.

$$C_D = C_{D0} + \frac{1}{\pi(AspectRatio)(OswaldEfficiency)} C_L^2 \quad (12-1)$$

Initially, C_{D0} was assumed to be 0.0020 from historical data. Based on the work in [101], C_{D0} can be better estimated by breaking down the drag into constituent components and calculating individual contributions. The drag breakdown for the GeoSurv UAV is shown in Table 12- 4 [101].

Table 12- 4: Parasite Drag Coefficient Drag Breakdown

Component	C_{D0} (based on wing reference area)	Percentage of Total (%)
Wing	0.0071	38
Fuselage	0.0042	22
Horizontal Tail	0.0007	3
Vertical Tail	0.0004	2
Trim Drag	0.00028	1
Wing Struts	0.00085	4
Landing Gear	0.0016	8
Tail Boom	0.0013	7
Interference Drag	0.00095	5
Cooling Drag	0.00095	5
Antenna Drag	0.00095	5
Total	0.019	100

From the revised C_{Do} , the equivalent parasite area can be calculated from Equation 12-2.

$$f = C_{Do} * \text{Wing Reference Area} \quad (12-2)$$

$$f = (0.019 * 250.98 \text{ ft}^2)$$

$$f = 4.8 \text{ ft}^2$$

The wetted area of the aircraft was calculated to be 945 ft² from the Geometric Configuration module. The Geometric Configuration module uses a series of correlations between major aircraft components and the individual Pro/Engineer solid models. When compared to the wetted area of the GeoSurv UAV measured from the combined Pro/Engineer solid model, there is less than a 5% difference at 985 ft².

Plotting of the values of equivalent parasite area and wetted area on Figure 12- 1 [102] gives a skin friction coefficient (C_f) of 0.0050. The skin friction coefficient then determines the coefficients used in the equivalent parasite area equation (12-3) used in the Aerodynamics module.

$$f = 10^{(a+b \log(\text{WettedArea}))} \quad (12-3)$$

If $C_f = 0.0050$, then $a = -2.3010$ and $b = 1.000$. The values of a and b were then updated in the Aerodynamics module.

Figure 12- 1: Effect of Equivalent Skin Friction and Wetted Area on Equivalent Parasite Area for Single Engine Propeller Driven Airplanes

12.1.5 Trial 2

The second trial involving the GeoSurv UAV was used to evaluate performance of the optimization algorithms on the MOCA problem formulation. The objective function was set to minimize the gross takeoff weight while ensuring that the performance constraints and variable bounds were met. An increasing number of independent variables were used to find algorithms that were capable of dealing with more and more complex optimizations.

Initially, only the thrust to weight ratio was used as an independent variable and nine different optimization algorithms or combination of algorithms were used, the results of which are visible in Table 12- 5. The initial design point is the first pass analysis through iSIGHT before the optimizations begin. The final design points are a result of the individual algorithms used, where the shaded cells indicate the best performance in each design variable category. The Directed Heuristic Search algorithm was not used for simplicity's sake; instead, the iSIGHT Advisor recommended algorithm was used.

Table 12- 5: Gross Takeoff Weight Variation with Optimization Algorithm and T/W

Optimization Algorithm	Initial Design Point	Advisor: NLPQL, MOST	NLPQL	MOST	LSCRG2	DONLP	Exterior Penalty	NNFD	SLP	Hooke-Jeeves Direct Search Method
Time to Completion	0 hr 1 min	1 hr 58 min	0 hr 30 min	2 hr 7 min	3 hr 25 min	0 hr 49 min	0 hr 45 min	0 hr 18 min	0 hr 46 min	0 hr 44 min
Total Number of Runs	1	136	35	149	249	58	52	18	49	52
Runs Per Minute	1.000	1.153	1.167	1.173	1.215	1.184	1.156	1.000	1.065	1.182
Run Number	1	114	21	125	100	20	12	17	10	39
Gross Takeoff Weight	3360.08	3274.34	3279.24	3274.34	3274.34	3313.79	3276.50	3274.52	3274.58	3274.36
Empty Weight	1740.51	1687.84	1692.01	1687.84	1687.84	1720.22	1689.61	1687.99	1688.03	1687.86
Fuel Weight	619.57	586.50	587.41	586.50	586.50	593.97	586.89	586.53	586.54	586.50
Aspect Ratio	9.18	9.18	9.18	9.18	9.18	9.18	9.18	9.18	9.18	9.18
Wing Span	48.00	48.00	48.00	48.00	48.00	48.00	48.00	48.00	48.00	48.00
Thrust To Weight Ratio	0.17993	0.16485	0.16674	0.16484	0.16484	0.18000	0.16564	0.16491	0.16493	0.16485
Wing Reference Area	250.98	250.98	250.98	250.98	250.98	250.98	250.98	250.98	250.98	250.98
Lift To Drag Ratio	14.258	14.946	14.946	14.946	14.946	14.946	14.946	14.946	14.946	14.946
Wing Loading	13.388	13.046	13.066	13.046	13.046	13.203	13.055	13.047	13.047	13.046
Power Loading	14.489	15.814	15.634	15.814	15.814	14.479	15.738	15.808	15.806	15.814
Comments [Note: All reported solutions are feasible. Comments indicate the progress of the algorithm through the design space during the optimization process.]		Found ok point quickly, then diverged into infeasible region with a few feasible worse.	Found ok point quickly, mostly feasible points found.	Found ok point, then diverged into infeasible region with occasional feasible points.	Closed in on ok value quickly, spent a lot of time on small changes, and still confused by tolerance on Wo. ∴ terminated by designer.	Centered around two points, one feasible, one not and made only very small changes. NOT GOOD	Found ok point quickly, but got stuck in infeasible region at end.	Found good point quickly.	Found good point very quickly, but then diverged into infeasible/feasible worse region for a long time.	Found good point quickly, consistent closing on value, but spent a lot of time on small changes at end.

The resulting aircraft is identical in size to the initial design point aircraft. The MOCA software tool also outputs a 3-D solid model in Pro/Engineer. Shown in Figure 12- 2, the GeoSurv UAV model initially generated by MOCA is very similar to the model created by the undergraduate project team, with the exception of the finer details that evolved during the preliminary design phase.

Figure 12- 2: Comparison of 3-D Conceptual Designs of the GeoSurv UAV

Figure 12- 3 shows a scaled version of both the project team (left) and the MOCA model (right). The maximum dimensions of the fuselage are identical, and the planforms are the same with the exception of minor differences in the size of the horizontal and vertical tails (due to placement). The noticeable differences include fuselage length, underbelly shape, boom length, boom angle, wing-strut size, and vertical tail leading edge sweep. The vertical tail leading edge sweep of the GeoSurv UAV project team model was changed after optimization runs had begun.

Figure 12- 3: Planform Comparison of the Initial GeoSurv UAV Models

Three optimization algorithms found the identical design optimum (i.e. the solution with the lowest gross takeoff weight), while the remaining algorithms were within 2% of the solution, indicating that the locally optimal GeoSurv UAV with a gross takeoff weight of

3274 lbs is likely a globally optimal solution after a thorough search of the design space. The MMFD algorithm found its optimal solution in the shortest number of runs, while LSGRG2 was the fastest on a per run basis and the SLP algorithm found its optimal solution earliest in the runs. Comments included in Table 12- 5 indicate qualitative performance details of individual algorithms.

Secondly, both T/W and wing span were varied. Only eight algorithms were used after examining the performance of the first set of optimization runs. DONLP was eliminated as a useful algorithm for large design spaces as it only searched around two major points, one feasible and one infeasible, neither of which were near the optimum found by the other algorithms. The performance of the second set of optimization runs can be seen in Table 12- 6, including comments of the performance of individual algorithms. Again, the MMFD algorithm found its optimal solution in the shortest number of runs and the LSGRG2 algorithm was the fastest. However, the MMFD algorithm found its optimal solution earliest in the runs. This is to be expected as the number of design variables increase and the other algorithms take significantly longer to find an optimal solution; the MMFD algorithm is consistently swift but less accurate than the others are. The best solution was found by the Hooke-Jeeves Direct Search Method, with a gross takeoff weight of 3267 lbs, down 2.8% from 3360 lbs. Compared to the first set of runs, this is only a decrease of 7 lbs in weight with a significant 25% decrease in wing span. The change in wing and empennage planforms can be seen in Figure 12- 4, with the light grey area showing the initial design and the dark grey overlay of the final, optimal design.

Table 12- 6: Gross Takeoff Weight Variation with Optimization Algorithm, T/W, and Wing Span

Optimization Algorithm	Initial Design Point	Advisor: NLPQL, MOST	NLPQL	MOST	LSCRG2	Exterior Penalty	MMFD	SLP	Hooke-Jeeves Direct Search Method
Time to Completion	0 hr 1 min	3 hr 48 min	1 hr 22 min	2 hr 41 min	3 hr 52 min	1 hr 13 min	0 hr 25 min	1 hr 2 min	1 hr 36 min
Total Number of Runs	1	268	96	186	279	85	25	72	113
Runs Per Minute	1.000	1.175	1.171	1.134	1.203	1.164	1.000	1.161	1.177
Run Number	1	265	83	183	236	83	21	71	26
Gross Takeoff Weight	3360.08	3277.30	3287.83	3274.44	3279.36	3303.37	3277.80	3270.02	3266.80
Empty Weight	1740.51	1691.35	1701.64	1687.96	1693.71	1715.59	1691.67	1676.80	1640.54
Fuel Weight	619.57	585.95	586.18	586.48	585.65	587.78	586.13	693.22	626.26
Aspect Ratio	9.18	9.18	9.18	9.18	9.18	9.18	9.18	9.18	9.18
Wing Span	48.00	48.52	49.363	48.02	48.86	50.13	48.48	44.93	36.00
Thrust To Weight Ratio	0.17993	0.16522	0.16792	0.16486	0.16550	0.17237	0.16546	0.16678	0.16868
Wing Reference Area	250.98	256.47	265.44	251.16	260.04	273.73	256.02	219.91	141.18
Lift To Drag Ratio	14.258	14.991	15.060	14.947	15.019	15.109	14.987	14.640	13.397
Wing Loading	13.388	12.779	12.386	13.037	12.611	12.068	12.803	14.870	23.140
Power Loading	14.489	15.778	15.525	15.813	15.752	15.124	15.756	15.631	15.455
Comments [Note: All reported solutions are feasible. Comments indicate the progress of the algorithm through the design space during the optimization process.]		Ok performance, didn't search full combination of lowered wing span and T/W.	Ok performance, didn't search full combination of lowered wing span and T/W.	Ok performance, went mostly infeasible and didn't vary wing span much.	Good performance, but did not reduce wing span or go above 49 ft.	Didn't try to reduce wing span or check far beyond initial point.	Good performance, but didn't try to reduce wing span.	Good performance, didn't try to reduce wing span until late.	Good performance, fairly thorough search, went infeasible at end.

Figure 12- 4: Planform Change As a Result of Wing Span Variation

Thirdly, T/W, wing span, and wing aspect ratio were used as independent variables. The performance of the third set of optimization runs is shown in Table 12- 7.

Table 12- 7: Gross Takeoff Weight Variation with Optimization Algorithm, T/W, Wing Span, and Aspect Ratio

Optimization Algorithm	Initial Design Point	Advisor: NLPQL, MOST	NLPQL	MOST	LSGRG2	Exterior Penalty	MMFD	SLP	Hooke-Jeeves Direct Search Method
Time to Completion	0 hr 1 min	4 hr 9 min	2 hr 9 min	2 hr 24 min	1 hr 10 min	2 hr 22 min	0 hr 34 min	1 hr 7 min	2 hr 38 min
Total Number of Runs	1	298	156	168	84	172	37	78	183
Runs Per Minute	1.000	1.197	1.209	1.167	1.200	1.211	1.088	1.164	1.158
Run Number	1	187	142	61	83	89	35	63	53
Gross Takeoff Weight	3360.08	3215.14	3181.08	3275.02	3273.43	3226.87	3267.69	3252.37	3205.20
Empty Weight	1740.51	1635.84	1631.18	1688.46	1687.35	1663.29	1684.54	1697.49	1649.90
Fuel Weight	619.57	579.31	549.90	586.56	586.07	563.57	583.15	554.88	555.30
Aspect Ratio	9.18	10.05	15.80	9.19	9.22	12.14	9.55	12.84	16.07
Wing Span	48.00	44.46	48.38	48.00	48.00	48.18	48.00	53.22	48.00
Thrust To Weight Ratio	0.17993	0.15199	0.14860	0.16516	0.16475	0.16060	0.16446	0.16667	0.15744
Wing Reference Area	250.98	196.66	148.13	250.73	249.97	191.17	241.18	220.66	143.42
Lift To Drag Ratio	14.258	14.803	15.846	14.949	14.957	15.579	15.037	16.197	15.788
Wing Loading	13.388	16.348	21.475	13.062	13.095	16.880	13.549	14.739	22.349
Power Loading	14.489	17.153	17.543	15.784	15.824	16.233	15.851	15.642	16.559
Comments [Note: All reported solutions are feasible. Comments indicate the progress of the algorithm through the design space during the optimization process.]		Poor start, went mostly infeasible. Didn't search far below start.	Started ok, went infeasible in middle.	Ok start, went infeasible and stayed.	Ok start, went infeasible quickly.	Good start, got stuck in infeasible region, tried to get out but didn't find better points. Didn't check very far from initial point except with aspect ratio.	Good start, but little refinement, and not a large range of variable values checked.	Good performance, went into infeasible region at end.	Started off well and then got stuck in infeasible region and stayed.

The MMFD algorithm found its optimal solution earliest and in the shortest number of runs, while the Exterior Penalty algorithm was the fastest on a per run basis. After examining the performance of each optimization algorithm operating on a single-objective problem formulation with up to three design variables, no one algorithm seems

to exhibit consistent, accurate performance. The best solution was found after comparing the results of all optimization algorithms and selecting the one with the lowest objective function value. The NLPQL algorithm found the best solution at a gross takeoff weight of 3181 lbs, down 5.3% from 3360 lbs, by increasing the wing span 0.8% and increasing the aspect ratio by 72.1%. The change in planform resulting from the optimization is illustrated in Figure 12- 5. It is important to note that the default settings for the basic/advanced parameters for algorithm tailoring were not modified in any of the optimizations. For simplicity, it was assumed that the defaults were satisfactory for initial comparison. The results suggest that this may not be the case for further studies.

Figure 12- 5: Planform Change As a Result of Wing Span and Aspect Ratio Variation

With the higher aspect ratio, the empennage of the aircraft is commensurately smaller. The fuselage could be shortened and the maximum fuselage diameter shrunk, while maintaining the balanced look of the aircraft. The gross takeoff weight of the aircraft at the preliminary build-up level would consequently decrease.

The T/W and, hence, power to weight ratio at takeoff for the optimal design is lower than that of the existing GeoSurv UAV. In combination with the smaller takeoff weight, this implies that a smaller, lighter engine could be used on the optimal design. The GeoSurv UAV currently utilizes a 230 hp engine at takeoff (based on a real engine), while the optimal design only requires 182 hp (based on a “rubber” engine). The problem arises when a real engine is sought for the optimal design in the preliminary design phase. An engine that provides 182 hp at takeoff that is smaller than the engine already employed may not exist. Thus, the implied weight savings of the optimal design with respect to the engine would disappear.

The smaller wing and empennage, and related drop in weight, comes at a high price in terms of high lift devices required to meet the performance constraints. Figure 12- 6 is a plot of the relevant performance matching inequality constraints with a high lift device setting of zero, and the infeasible design space shown grey hatching. The feasible design space is shown in white. The sea level stall speed constraint is only satisfied if the wing has a maximum lift coefficient of approximately 3.4, corresponding to triple slotted flaps plus slats. The existing GeoSurv UAV only employs Fowler flaps. To switch to such a complex set of high lift devices would increase the complexity of the manufacturing and

the required autopilot control, as well as the total weight of the high lift device systems. Since individual components are not estimated at this phase in the design, there is a high probability that the weight of the complex high lift devices would cancel out most of the effect of reducing the wing and empennage size on the gross takeoff weight. As a result, the “optimal” design may not actually be the optimal solution when preliminary design considerations are taken into account.

Figure 12- 6: Performance Matching for the Optimal Design Point with Triple Slotted Flaps plus Slats for the Sea Level Stall Condition

Also important to note in Figure 12- 6 is the location of the design point slightly below the sea level climb constraint, in the infeasible region. The Performance Matching

module returned the small constraint violation to iSIGHT, but the design point was considered within the accuracy of the of the optimization algorithm and deemed feasible. However, even a small constraint violation means that the design will not be able to meet the mission requirements. The accuracy and magnitude of the measure of the constraint violation will need to be examined further.

12.2 Test Case 2

The second test case is an existing unmanned aircraft with proven performance characteristics, the I-GNAT by General Atomics Aeronautical Systems, Inc. The sample mission requirements and geometric configuration were provided in advance of the name and actual gross takeoff weight of the aircraft as a blind test to see how similar the results would be. In addition, the estimated gross takeoff weight and wing span were modified to determine the effect of changing the initial design point on the optimal design.

12.2.1 Sample Mission Requirements

The exact mission requirements that the I-GNAT was designed for are not known. The sample mission requirements, shown below, were developed from information published by the manufacturer about the aircraft [103].

Endurance:	40 hours
Range:	50 nm
Cruise Speed:	80 kts

Payload Weight:	200 lbs
Sea Level Stall Speed:	60 kts
Cruise Altitude:	10,000 ft
Takeoff and Landing Field Length:	2000 ft, ISA
Actual Gross Takeoff Weight:	1550 lbs

12.2.2 Geometric Configuration

The I-GNAT is a long endurance, medium altitude, pusher-propeller aircraft. It has a composite airframe, and a conic shape best approximates the fuselage. The wing is mounted low on the fuselage, the empennage is a V-shape tail mounted below the fuselage, and the landing gear is retractable. Refer to Table 12- 8 for a summary of the geometric configuration options input into the MOCA software tool. Table 12- 9 lists the values of the geometric configuration parameters.

Table 12- 8: I-GNAT Geometric Configuration Options

Variable	Value
Aircraft Material Type	Composite
Fuselage Symmetry	Symmetric
Fuselage Cross-Section Shape	Conic
Wing Placement	Low Wing
Canard Type	No Canard
Horizontal Tail Type	None
Vertical Tail Type	V-Shape
Engine Type	Propeller
Engine Placement	Fuselage, Internal
Engine Direction	Pusher
Landing Gear/Wing Configuration Type	Retractable, Cantilevered Wing

Table 12- 9: I-GNAT Geometric Configuration Parameter Values

Variable	Value
Fuselage Number	1
Maximum Fuselage Diameter	3.0
Fuselage Aspect Ratio	1.5
Wing Span	38.00
Wing Aspect Ratio	13.103
Wing Root Chord	4.0
Wing Tip Chord (Starboard and Port)	1.8
Wing Oblique Angle	0
Wing Dihedral 1-6	0
Wing Incidence	0
Wing Twist 1-6	0
Wing Leading Edge Sweep 1-6	0
Wing Taper Ratio 1	1.00
Wing Taper Ratio 2	0.45
Wing Taper Ratio 3	1.00
Wing Taper Ratio 4	1.00
Wing Taper Ratio 5	0.45
Wing Taper Ratio 6	1.00
Number of Wing Breakpoints	0
Wing Breakpoint Y-Distance 1,4	19.00
Wing Breakpoint Y-Distance 2,3	0.00
Vertical Tail 1 Rotation Angle	315
Vertical Tail 2 Rotation Angle	225
Vertical Tail 1,2 Span	4.12
Vertical Tail 1,2 Aspect Ratio	2.75
Vertical Tail 1,2 Leading Edge Sweep	0
Vertical Tail 1,2 Taper Ratio	1
Vertical Tail 1,2 Incidence	0
Vertical Tail 1,2 Twist	0
Number of Booms	0
Number of Engines	1
Gross Takeoff Weight Estimate	2500

12.2.3 Trial 1

From the results of the first test case, the MMFD optimization algorithm was able to obtain a near optimal solution, within six thousandths of a percent. The MMFD algorithm also found the solution in less than one sixth of the time, as compared to other algorithms that obtained optimal or near optimal solutions, when only T/W was varied. The MMFD algorithm was used to find the minimum gross takeoff weight of the I-GNAT to minimize the time taken per optimization run. The skin friction coefficient was then varied to determine the effect on the gross takeoff weight, as no drag information was available on the aircraft. Table 12- 10 presents the findings.

Table 12- 10: Gross Takeoff Weight Variation with C_f and T/W

Skin Friction Coefficient	Initial Design Point $C_{Do} = 0.020$	0.0020	0.0030	0.0040	0.0050
Time to Completion	0 hr 1 min	0 hr 30 min	0 hr 51 min	0 hr 33 min	0 hr 21 min
Total Number of Runs	1	35	47	28	22
Runs Per Minute	1.000	1.167	0.922	0.848	1.048
Run Number	1	31	42	17	9
Gross Takeoff Weight	1787.02	921.41	1067.12	1259.46	1511.80
Empty Weight	898.15	468.66	530.20	619.92	737.49
Fuel Weight	688.87	252.75	336.92	439.54	574.31
Aspect Ratio	13.103	13.103	13.103	13.103	13.103
Wing Span	38.00	38.00	38.00	38.00	38.00
Thrust To Weight Ratio	0.15844	0.06610	0.06518	0.07609	0.09148
Wing Reference Area	110.20	110.20	110.20	110.20	110.20
Lift To Drag Ratio	16.190	26.533	21.628	18.709	16.540
Wing Loading	16.216	8.361	9.683	11.428	13.718
Power Loading	20.567	49.300	49.996	42.827	35.620
High Lift Device Setting	5	5	5	5	5

From the shaded row in Table 12- 10, it is evident that the gross takeoff weight is quite sensitive to the skin friction coefficient, which is used to determine the lift-to-drag ratio of the aircraft. As C_f increases, L/D decreases and the weight of the aircraft rises. Consequently, it is important to have an accurate estimate of the aircraft drag to be able to determine the gross takeoff weight.

From Table 12- 10, it can also be seen that the high lift device setting is always reported at the highest level (out of a possible 0 through 5) indicating slats and triple slotted flaps. The Performance Matching module database was examined, and unlike the GeoSurv UAV test case, the amount of constraint violation was not changing for each of the high lift device settings. Figure 12- 7 plots the performance constraints with a high lift device setting of zero (i.e. clean configuration) and in comparison, Figure 12- 8 is plotted with a high lift device setting of five. The design point plotted on each figure is the optimal point found from Table 12- 10 with $C_f = 0.0030$. Since the design point is clearly within the feasible design space at both high lift device settings, the lower setting should have been chosen. Since it was not, all of the high lift device settings are suspect and the discrepancy will have to be evaluated further. However, the constraint violation from the Performance Matching module is returned to the Analysis Modules optimization correctly and the performance of the higher-level optimization remains unaffected.

**Figure 12- 7: Performance Matching for the Optimal Design Point at $C_f = 0.0030$
and No Flaps for the Sea Level Stall Condition**

**Figure 12- 8: Performance Matching for the Optimal Design Point at $C_f = 0.0030$
and Triple Slotted Flaps plus Slats for the Sea Level Stall Condition**

When the T/W values from Table 12- 10 are examined, the optimal values are significantly lower than both the initial design point and the existing aircraft design point. When the existing I-GNAT UAV design is plotted on a performance matching graph with no flaps at the sea level stall condition, Figure 12- 9 results. Here, the design point is much higher on the plot and further away from the constraint boundaries. There are two reasons for the high T/W value. One, the farther the design point is from the constraint boundaries, the better the design under a range of operating conditions. The resulting design is far more flexible allowing the manufacturer to offer an aircraft that suits a greater number of applications and increasing sales. Two, the selection of engines for the real world application is limited to a small set of maximum takeoff horsepower values. The existing engine may have been chosen because it was the lightest engine that met the minimum performance requirements.

Figure 12- 9: Performance Matching for the I-GNAT UAV Design Point

In addition, the gross takeoff weight was arbitrarily set to 2500 lbs, then 1000 lbs and 4500 lbs, to determine the effect on the start of the optimization. No observable differences were noted in the initial design point, which indicates that the initial sizing module is robust.

The Pro/Engineer models resulting from the second trial are identical to the initial design point, as the skin friction coefficient has no effect on the geometric configuration parameters. Figure 12- 10 shows the model resulting from MOCA. The MOCA model is essentially the same configuration as the existing I-GNAT, but differences exist in fuselage size and shape. Figure 12- 11 shows the I-GNAT UAV in flight for comparison.

Figure 12- 10: The Initial Solid Model of the I-GNAT UAV

Figure 12- 11: The I-GNAT UAV

In Figure 12- 12, the rear of the aircraft is illustrated with a conceptual engine clearly protruding from the fuselage. The tail cone is too long and slender to house a piston engine appropriately.

Figure 12- 12: Engine Fit in Initial Fuselage

By altering the main fuselage parameters in the Geometric Configuration module (without re-running the optimization) as shown in Table 12- 11, the Pro/Engineer solid model can be modified to provide a realistic tail shape to accommodate the pusher-propeller engine.

Table 12- 11: Modified Fuselage Parameters

	Default Values	Modified Values
Nose Fineness Ratio	2.00	1.75
Constant Cross-Section Fineness Ratio	3.00	3.00
Tail Fineness Ratio	5.00	1.75
Tail Angle	10.0°	1.00°

Figure 12- 13: Engine Fit in Modified Fuselage

Figure 12- 13 demonstrates how the modified fuselage fits most of the engine in height, width, and length, with only minor modifications required to house the portion of the engine protruding from the fuselage. The engine cowling is quite blunt but at the conceptual stage, the engine cowling would not have to be re-designed. Only later in the design phases, once an engine is selected and major dimensions are known, would the exact cowling shape be determined. Figure 12- 14 illustrates the effect of modifying the main fuselage parameters to resemble the actual aircraft more closely.

Figure 12- 14: The Modified Fuselage Solid Model of the I-GNAT UAV

An attempt was also made in the second test case to determine the effect of changing the initial design point on the optimal design. The wing span was entered as 38 ft, as opposed to the actual value of 42 ft, to see what the optimal wing span would be. Because the individual algorithms are sensitive to the initial design point and the number of design variables, testing would have to consider both. Unfortunately, the effect of the start point was too complex to determine in the available time.

13. Conclusions

The purpose of this thesis was to begin the development of a software tool for the multidisciplinary optimization of conceptual aircraft design to address issues that arose with older MDO tools. The first development stage of the software tool was completed and its performance was evaluated on two aircraft designs, one in the preliminary design phase and the other already in use.

13.1 Current Software Development

The following is a direct assessment of the MOCA software tool against the criteria outlined in Section 7. The MOCA software tool:

- is user-friendly. The GUI is straightforward and easy to use.
- is of industrial strength. iSIGHT can handle a large number of design variables and constraints.
- is not yet fully robust. Convergence is only attained under a wide range of gross takeoff weight values.
- is efficient, for reasonable levels of computing time. A full optimization may take between 10 minutes and 4 hours on a personal computer in a Windows XP Professional environment.
- is reconfigurable. Nothing prevents the designer from adding formalized routines, substituting judgement for a formal method, or overriding results within the confines of the input and output requirements of the optimization framework.

Provisions also exist for a list of appropriate actions to be taken by designer if deviating from standard procedure: refer to [12].

- has a logical, structured, modular design. This aids in reconfiguration and enhances the program's extensibility.
- does not select what it needs to calculate based on the supplied information and objective function requirements. This functionality is currently beyond the scope of the project.
- has a flexible method of data entry, with as little required data as possible. Effort was made to minimize the amount of input variables required.
- does not yet utilize relational databases, due to time limitations.
- has the ability to accommodate a wide variety of informational sources such as estimates, statistics, references, experiments, look up tables, and other software. This is inherent in the use of Matlab for analysis and iSIGHT as the optimization framework.
- does not use many start points to explore the design space fully. Cold, warm, and hot starts are not yet supported.
- supports interdigitation. This is a feature of iSIGHT.
- inspires confidence in optimization results by allowing the designer to view the analysis and optimization process. The designer can interrupt the process at any point.
- has a fast, comprehensive, easy-to-interpret output format that fulfills previously determined visualization requirements.

- has limited support for interactive design changes to assess impact. The Pro/Engineer models are fast and easy to modify, but the effect of design changes can only be examined through re-optimization.
- provides full documentation (refer to [12]); help files have not yet been developed.
- supports transition of generated data to preliminary design models and codes through the use of Pro/Engineer.

As for requirements that were specifically developed for conceptual aircraft design optimization software, the MOCA software tool:

- is set up to be able to handle a wide variety of aircraft types. At this time, only unmanned vehicles, in both conventional and unconventional configurations, are supported. Flying wings are not yet supported.
- uses a generic geometric aircraft representation, implemented with Pro/Engineer.
- can optimize an existing aircraft design. Due to complexity and time constraints, the ability to generate conceptual designs from mission requirements is not yet supported.
- cannot yet support multiple flight conditions, but the mission profile is quite flexible.
- has the ability to use alternate modules for different types of aircraft/analyses. Alternate modules have not yet been implemented.

13.2 Selected Software Performance

iSIGHT, the optimization framework, was the fundamental tool that provided the MOCA program with flexibility, ease of learning and use, a high degree of designer control, and a transparent optimization process. Additional software analysis packages were not as easy to integrate as training tutorials indicated. However, once the procedures were established (refer to Appendices C and D) the simcode calls became routine. The ability to generate description files dynamically was critical to the development of the designer input GUI. Regrettably, an effective solution had not been found at the time of the completion of the thesis to change the default sixty-second time limit on information input into the GUI.

Matlab, as the primary analysis environment, was suitable to the tasks required of it. The use of individual modules written in separate M-files aided in the modularity of the software structure. Matlab evaluated large amounts of code faster than the calculations in iSIGHT, but it was slow to initialize. This was a direct result of having to open and close Matlab on every simcode call.

Pro/Engineer was used for artifact-based visualization purposes in the MOCA program. The trail files are exacting in the conditions required to run correctly, but are easy to integrate into the optimization framework. The solid models of the optimal aircraft designs are useful both to the designer at the conceptual design phase, and to future development in the preliminary phase.

13.3 Software Tool Performance

The performance of the MOCA software tool was evaluated using two different aircraft in differing stages of development. The first test case was used to compare the results of common hand calculations to the analysis modules within iSIGHT. Given the small variations in the analyses discussed in each of the approaches, the calculations performed in MOCA provided values quite similar to both sets of hand calculations; therefore, the MOCA analysis modules are assumed correct. Secondly, the performance of the optimization algorithm on the given problem formulation was evaluated. The larger the number of design variables used in the optimization, the less the algorithms agreed on the optimal solution. No one algorithm, with default parameter settings, seems to have best performance in accuracy, but the MMFD algorithm was consistently the fastest to complete an optimization. Further research is required to tailor the basic/advanced parameters to the problem formulation in MOCA to determine if one algorithm is clearly better than all the others. The performance of the constraint violation parameter in the Performance Matching module must be assessed in both accuracy and magnitude. Additional study is also required in interdigitation, as a combination of algorithms may be more effective than a single algorithm.

The second test case was completed blind to see how the results would compare to the existing aircraft. Numerically, the MOCA aircraft was much lower in T/W and gross takeoff weight values due to the location of the design point with respect to the performance matching constraints. Physically, the MOCA results produced a highly

similar aircraft after the Pro/Engineer model was modified slightly. The analysis modules were found to have good robustness with respect to the entered estimate of gross takeoff weight, but a significant sensitivity to the skin friction coefficient estimate for the aircraft. An issue arose with the selection of the high lift device setting in the Performance Matching module that must be examined. The engine sizing and placement were found to be suitable for only the conceptual design phase. In addition, investigation into the effect of changing the initial design point on the optimal design was started. However, the scope was too large to cover in the available time.

13.4 Further Software Development

In the context of unmanned aerial vehicles, more development is required in the geometric configuration to be able to accommodate flying wings/blended wing-body configurations, wing tip modifications (such as winglets), engine shapes and ducting, landing gear, and control surfaces and high lift devices. Additional analysis modules should then be included (refer to Figure 8- 3), and after which, manned aircraft design can be started.

In terms of optimization work, the selection of an objective function based on considerations that are more realistic (especially in terms of compound objectives) is a high priority. Additional system constraints and optimization algorithm performance must also be examined.

14. Future Project Development

The tasks accomplished in the first software development stage are but the first steps towards completion of the MOCA program. The designer should bear in mind that the solutions provided by MOCA are the result of an incomplete software tool. As new analysis modules are developed, the solutions found by MOCA will be more accurate and more likely to be feasible considering all of the constraints on the system. As with any design tool, the MOCA program should not simply be used as a “black box” with absolute certainty in the output.

The remaining tasks for the program include:

- 1) Investigation into the lack of change of constraint violation in the Performance Matching module and the selection of high lift device setting.
- 2) Modification of the Aerodynamics module to include the skin friction coefficient as an input, or the addition of a drag breakdown method to determine a more accurate skin friction coefficient.
- 3) Reconfiguration of the input GUI and analysis modules to accommodate multiple mission profiles.
- 4) Development of the Geometric Configuration module to be able to accommodate flying wings/blended wing-body configurations, wing tip modifications, various engine shapes and ducting, landing gear, and control surfaces and high lift devices.

- 5) Instituting a more sophisticated method of calculating the center of gravity of the aircraft.
- 6) Completion of the remaining Level 1 modules for unmanned aerial vehicles.
- 7) Development of the historical database and addition of data from Jane's All the World's Aircraft.
- 8) Extension of the historical database to include 3-D models, either using the Geometric Configuration module to generate the design variables from the information provided in the database or by utilizing the advancements made in computer aided-drafting programs to convert 2-D sketches into 3-D models.
- 9) Completion of initial design templates and Level 1 modules for other aircraft types (applicable modules may be reused).
- 10) Extension of the analyses to include Level 2 and 3 type calculations and/or integrated software.
- 11) Extension of the programming structure to include preliminary design phase computations.

The following performance conditions must also be evaluated:

- 1) Testing of different aircraft configuration types, and individual optimization algorithm performance on this broader base of test cases.
- 2) Tailoring of the basic/advanced parameters of individual algorithms to the problem formulation in MOCA to determine if one algorithm is clearly better than all the others.
- 3) Tailoring of variable scaling to determine the effect on finding the optimal design.
- 4) The study of various interdigitation methods, which may be more effective than a single algorithm.
- 5) The effect of changing the initial design point on the optimal design.
- 6) The selection of an objective function based on realistic, multiobjective considerations.
- 7) Additional system constraints to develop a more realistic model of the aircraft as a system.

REFERENCES

- [1] Sobieszczanski-Sobieski, J., and Haftka, R. T., 1997, "Multidisciplinary Aerospace Design Optimization: A Survey of Recent Developments," *Structural Optimization*, **14**, pp.1-23.
- [2] Giesing, J. P. and Barthelemy, J.-F. M., 1998, "A Summary of Industry MDO Applications and Needs," AIAA Summary White Paper 98-4737,
http://adg.stanford.edu/aa222/giesingsummary_paper.pdf
- [3] AIAA Technical Committee on Multidisciplinary Design Optimization, 1991,
"White Paper on Current State of the Art," American Institute of Aeronautics and Astronautics, http://endo.sandia.gov/AIAA_MDOTC/sponsored/aiaa_paper.html
- [4] Mah, S., 2001, "PMDO Preliminary Multidisciplinary Design Optimizer,"
Presentation for the AIAC Technology Collaboration Forum 03 May 2001, Pratt & Whitney Canada.
- [5] Bil, C., 1988, "Development and Application of a Computer-Based System for Conceptual Aircraft Design," Ph.D. Thesis, Delft University Press.
- [6] Torenbeek, E., 1988, *Synthesis of Subsonic Airplane Design*, Delft University Press, Delft, Netherlands.
- [7] Olds, J. R., 1992, "The Suitability of Selected Multidisciplinary Design and Optimization Techniques to Conceptual Aerospace Vehicle Design," AIAA Paper 92-4791, *4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Cleveland, OH, pp. 820-843.

- [8] Nicolai, L. M., and Carty, A., 1999, "Role of the Aerodynamicist in a Concurrent Multi-disciplinary Design Process," Lockheed Martin Skunk Works.
- [9] Hajela, P., 2000, "Optimal Design in Multidisciplinary Systems," AIAA Professional Study Series, Course Outline.
- [10] Kroo, I. M., and Takai, M., 1988, "A Quasi-Procedural, Knowledge-Based System for Aircraft Design," AIAA Paper 88-6502.
- [11] Roskam, J., 1990, Airplane Design Part VIII, Airplane Cost Estimation: Design, Development, Manufacturing and Operating, Roskam Aviation and Engineering Corporation, Ottawa, KS, pp. 10, Chap. 2.
- [12] Eisler, C. A., 2004, "MOCA Software Documentation," Carleton University, Ottawa, ON.
- [13] Malone, B., and Mason, W. H., 1992, "Aircraft Concept Optimization Using the Global Sensitivity Approach and Parametric Multiobjective Figures of Merit," AIAA Paper 92-4221.
- [14] Dovi, A. R. and Wrenn, G. A., 1990, "Aircraft Design for Mission Performance Using Nonlinear Multiobjective Optimization Methods," Journal of Aircraft, **27**(12), pp. 1043-1049.
- [15] Messac, A., 2000, "From the Dubious Construction of Objective Functions to the Application of Physical Programming," AIAA Journal, **38**(1), pp. 155-163.
- [16] Messac, A. and Wilson, B. H., 1998, "Physical Programming for Computational Control," AIAA Journal, **36**(2), pp. 219-226.

- [17] Messac, A. and Chen, X., 2000, "Visualizing the Optimization Process in Real-Time Using Physical Programming," Rensselaer Polytechnic Institute, Troy, NY, <http://www.rpi.edu/~messac/Publications/visualization.pdf>
- [18] Chinneck, J. W., 2001, 94.504 Mathematical Programming for Engineering Applications: Course Notes, Carleton University, Ottawa, ON.
- [19] Winston, W. L., 1995, *Introduction to Mathematical Programming: Applications and Algorithms*, Duxbury Press, Belmont, CA, pp.751, Chap. 12.
- [20] Sobieszczanski-Sobieski, J., 1995, "Multidisciplinary Design Optimization: An Emerging, New Engineering Discipline," *Advances in Structural Optimization*, J. Herskovitz, ed., Kluwer Academic Publishers, Dordrecht, pp. 483-496.
- [21] Kroo, I. M., 2001, "Optimization and Design (Introduction)," AA222 - Multidisciplinary Design Optimization Course Material, <http://adg.stanford.edu/aa222/optimizationanddesign.html>
- [22] Daberkow, D.D. and Mavris, D. N., 1998, "New Approaches to Conceptual and Preliminary Aircraft Design: A Comparative Assessment of a Neural Network Formulation and a Response Surface Methodology," AIAA Paper 98-5509.
- [23] Toropov, V. V. and Markine, V. L., 1996, "The Use of Simplified Numerical Models as Mid-range Approximations," AIAA Paper 96-4088, *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, pp. 952-958.
- [24] Vanderplaats, G. N., 2000, "Approximation Techniques," Presentation Notes, Vanderplaats Research and Development, Inc., Colorado Springs, CO.

- [25] Arora, J. S. and Belegundu, A. D., 1984, "Problems in Large-Scale Structural Optimization," *Recent Experiences in Multidisciplinary Analysis and Optimization*, NASA CP-2327, Hampton, VA, pp. 847-872.
- [26] Frank, P. D., Booker, A. J., Caudell, T. P., and Healy, M. J., 1992, "A Comparison of Optimization and Search Methods for Multidisciplinary Design," AIAA Paper 92-4827, *4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Cleveland, OH, pp. 1040-1057.
- [27] Raymer, D. P. and Crossley, W. A., 2002, "A Comparative Study of Genetic Algorithm and Orthogonal Steepest Descent for Aircraft MDO," AIAA Paper 2002-0514.
- [28] Unal, R., Lepsch, R. A., Englund, W., and Stanley, D. O., 1996, "Approximation Model Building and Multidisciplinary Design Optimization Using Response Surface Methods," AIAA Paper 96-4044, *6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, pp. 592-598.
- [29] Crispin, Y., 1992, "Adaptive Optimization of General Aviation Aircraft," AIAA Paper 92-4195.
- [30] Martins, J., 2001, "Sensitivity Analysis," Course Material for AA222 - Multidisciplinary Design Optimization, Stanford University,
<http://adg.stanford.edu/aa222/sensitivity.pdf>
- [31] Sobieszczanski-Sobieski, J., 1990, "Sensitivity Analysis and Multidisciplinary Optimization for Aircraft Design: recent Advances and Results," *Journal of Aircraft*, **27**(12), pp. 993-1001.

- [32] Braun, R. D., Kroo, I. M., and Gage, P., 1993, "Post-Optimality Analysis in Aerospace Vehicle Design, AIAA Paper 93-3932.
- [33] Sobieszczanski-Sobieski, J., Barthelemy, J.-F. M., and Riley, K. M., 1981, "Sensitivity of Optimum Solution to Problem Parameters," AIAA Paper 81-0548.
- [34] Barthelemy, J.-F. M. and Sobieszczanski-Sobieski, J., 1983, "Optimum Sensitivity Derivatives of Objective Functions in Nonlinear Programming," AIAA Journal, **21**(6), pp. 913-915.
- [35] Barthelemy, J.-F. M. and Sobieszczanski-Sobieski, J., 1983, "Extrapolation of Optimum Design Based on Sensitivity Derivatives," AIAA Journal, **21**(5), pp. 797-799.
- [36] Sobieszczanski-Sobieski, J., 1990, "Sensitivity of Complex, Internally Coupled Systems," AIAA Journal, **28**(1), pp. 153-160.
- [37] Hajela, P., Bloebaum, C. L., and Sobieszczanski-Sobieski, J., 1990, "Application of Global Sensitivity Equations in Multidisciplinary Aircraft Synthesis, Journal of Aircraft, **27**(12), pp. 1002-1010.
- [38] Sobieszczanski-Sobieski, J., 1990, "Higher Order Sensitivity Analysis of Complex, Coupled Systems," AIAA Journal, **28**(4), pp. 756-758.
- [39] Wagner, T. C. and Papalambros, P. Y., 1993, "A General Framework for Decomposition Analysis in Optimal Design," Advances in Design Automation, **2**, pp. 315-325.
- [40] Balling, R. J. and Sobieszczanski-Sobieski, J., 1994, "Optimization of Coupled Systems: A Critical Overview of Approaches," AIAA Paper 94-4330, *5th*

AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City Beach, FL, pp. 753-773.

- [41] Braun, R. D., Gage, P., Kroo, I. M., and Sobieski, I., 1996, "Implementation and Performance Issues in Collaborative Optimization," AIAA Paper 96-4017.
- [42] Sobieszczanski-Sobieski, J., 1988, "Optimization by Decomposition: A Step from Hierarchic to Non-Hierarchic Systems," NASA CP-3031, Recent Advances in Multidisciplinary Analysis and Optimization, J.-F. M. Barthelemy, ed., Hampton, VA, pp. 51-78.
- [43] Logan, T. R., 1990, "Strategy for Multilevel Optimization of Aircraft," *Journal of Aircraft*, **27**(12), pp. 1068-1072.
- [44] Eason, E. D. and Wright, J. E., 1992, "Implementation of Non-Hierarchic Decomposition for Multidisciplinary System Optimization," AIAA Paper 92-4822, *4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Cleveland, OH, pp. 1008-1016.
- [45] Eason, E. D., Nystrom, G. A., Burlingham, A., and Nelson, E. E., 1994, "Non-Hierarchic Multidisciplinary Design of a Commercial Aircraft," AIAA Paper 94-4302, *5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, pp. 493-504.
- [46] Renaud, J. E. and Gabriele, G. A., 1994, "Approximation in Nonhierarchic System Optimization," *AIAA Journal*, **32**(1), pp. 198-205.
- [47] Wujek, B. A., Renaud, J. E., Batill, S. M., and Brockman, J. B., 1995, "Concurrent Subspace Optimization Using Design Variable Sharing in a Distributed Computing

- Environment,” *Proceedings of the 1995 Design Engineering Technical Conference*, American Society of Mechanical Engineers, New York, NY, Vol. 82, pp. 181-188.
- [48] Wujek, B. A., Renaud, J. E., Batill, S. M., Johnson, E. W., and Brockman, J. B., 1996, “Design Flow Management and Multidisciplinary Design Optimization in Application to Aircraft Concept Sizing,” AIAA Paper 96-0713.
- [49] Kroo, I. M., Altus, S., Braun, R. D., Gage, P., and Sobieski, I., 1994, “Multidisciplinary Optimization Methods for Aircraft Preliminary Design,” AIAA Paper 94-4325.
- [50] Kroo, I. M. and Manning, V., 2000, “Collaborative Optimization: Status and Directions,” AIAA Paper 2000-4721, *8th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach CA.
- [51] Sobieski, I. and Kroo, I. M., 1996, “Aircraft Design Using Collaborative Optimization,” AIAA Paper 96-0715.
- [52] Sobieszczanski-Sobieski, J., Agte, J., and Sandusky Jr., R., 1998, “Bi-Level Integrated System Synthesis,” AIAA Paper 98-4916.
- [53] Sobieszczanski-Sobieski, J., Emiley, M. S., Agte, J., and Sandusky Jr., R., 2000, “Advancement of Bi-level Integrated System Synthesis (BLISS),” AIAA Paper 2000-0421.
- [54] Rogers, J. L., 1999, “Tools and Techniques for Decomposing and Managing Complex Design Projects,” *Journal of Aircraft*, **36**(1), pp. 266-274.
- [55] Rogers, J. L., 1996, “DeMAID/GA - An Enhanced Design Manager's Aid for Intelligent Decomposition,” AIAA Paper 96-4157.

- [56] Rogers, J. L., 1989, "A Knowledge-Based Tool for Multilevel Decomposition of a Complex Design Problem," NASA TP-2903, NASA Langley Research Center, Hampton, VA.
- [57] Sues, R. H., Oakley, D. R., and Rhodes, G. S., 1996, "MDO of Aeropropulsion Components Considering Uncertainty," AIAA Paper 96-4062.
- [58] Du, X., and Chen, W., 2001, "A Hierarchical Approach to Collaborative Multidisciplinary Robust Design," *4th World Congress of Structural and Multidisciplinary Optimization*, Dalian, China,
www.uic.edu/labs/idel/pdf/WCSMO01.pdf
- [59] Goel, A., Baker, C. A., Shaffer, C. A., Grossman, B., Haftka, R. T., Mason, W.H., and Watson, L.T., 1999, "VizCraft: A Multidimensional Visualization Tool for Aircraft Configuration Design," MAD Center Report 99-06-01, Virginia Polytechnic Institute and State University, Blacksburg, VA,
http://www.aoe.vt.edu/aoe/faculty/Mason_f/MAD990601.pdf
- [60] Raymer, D. P., 2001, "Vehicle Scaling Laws for Multidisciplinary Optimization" AIAA Aerospace Sciences Meeting, Reno, NV.
- [61] Bartholomew, P., 1998, "The Role of MDO Within Aerospace Design and Progress Towards an MDO Capability", AIAA Paper 98-4705, *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO.
- [62] Smith, R. E. and Kerr, P. A., 1992, "Geometric Requirements for Multidisciplinary Analysis of Aerospace Vehicle Design," AIAA Paper 92-4773, *4th*

AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, pp. 686-694.

- [63] Smith, R. E., Bloor, M. I. G., Wilson, M. J., and Thomas, A. T., 1995, "Rapid Airplane Parametric Input Design (RAPID)," AIAA Paper 95-1687.
- [64] Johnson, V., 1990, "Minimizing Life Cycle Cost for Subsonic Commercial Aircraft," *Journal of Aircraft*, **27**(2), pp. 139-145.
- [65] Cousin, J. and Metcalfe, M., 1990, "The BAe (Commercial Aircraft) Ltd. Transport Aircraft Synthesis and Optimization Program," AIAA Paper 90-3295.
- [66] Bao, H. P. and Samareh, J. A., 2000, "Affordable Design: A Methodology to Implement Process-Based Manufacturing Cost Models into the Traditional Performance-Focused Multidisciplinary Design Optimization," AIAA Paper 2000-4839.
- [67] Rais-Rohani, M. and Dean, E. B., 1996, "Toward Manufacturing and Cost Considerations in Multidisciplinary Aircraft Design," AIAA Paper 96-1620, *37th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Salt Lake City, UT, pp. 2602-2612.
- [68] DARcorporation, 2003, "Advanced Aircraft Analysis (AAA)," Lawrence, KS, <http://www.darcorp.com/Software/aaa.htm>
- [69] Roskam, J. and Malaeks, S., 1989, "Automated Aircraft Configuration Design and Analysis," SAE Technical Paper 891072.
- [70] Roskam, J. and Anemaat, W. A., 1996, "General Aviation Aircraft Design Methodology in a PC Environment," AIAA Paper 96-5520.

- [71] Raymer, D. P., 1992, "RDS: A PC-Based Aircraft Design, Sizing and Performance System," AIAA Paper 92-4226.
- [72] Myklebust, A. and Gelhausen, P., 1994, "Putting the ACSYNT on Aircraft Design," *Aerospace America*, **32** (9), pp. 26-30.
- [73] Jayaram, S., Myklebust, A., and Gelhausen, P., 1992, "ACSYNT - A Standards-Based System for Parametric Computer Aided Conceptual Design of Aircraft," AIAA Paper 92-1268.
- [74] Hale, M. A., Mavris, D. N., and Carter, D. L., 1999, "The Implementation of a Conceptual Aerospace Systems Design and Analysis Toolkit," AIAA Paper 99-5639.
- [75] Arbuckle, D. and Sliwa, S. M., 1983, "Parametric Study of Critical Constraints for a Canard Configured Medium Range Transport Using Conceptual Design Optimization," AIAA Paper 83-2141.
- [76] Sliwa, S. M., 1980, "Use of Constrained Optimization in the Conceptual Design of a Medium-Range Subsonic Transport," NASA TP-1762, NASA Langley Research Center, Hampton, VA.
- [77] Arbuckle, D. and Sliwa, S. M., 1985, "Parametric Study of a Canard-Configured Transport Using Conceptual Design Optimization," NASA TP-2400, Hampton, VA.
- [78] Arbuckle, D. and Sliwa, S. M., 1984, "Experiences Performing Conceptual Design Optimization of Transport Aircraft," *Recent Experiences in Multidisciplinary Analysis and Optimization*, NASA CP-2327, Hampton, VA, pp. 87-101.

- [79] McCullers, L. A., 1984, "Aircraft Configuration Optimization Including Optimized Flight Profiles," *Recent Experiences in Multidisciplinary Analysis and Optimization*, NASA CP-2327, Hampton, Virginia, **Part 1**, pp. 395-412.
- [80] Gabriele, G. A., 1984, "Application of the Generalized Reduced Gradient Method to Conceptual Aircraft Design," *Recent Experiences in Multidisciplinary Analysis and Optimization*, NASA CP-2327, Hampton, VA, pp. 65-85.
- [81] Simos, D. and Jenkinson, L. R., 1986, "Optimisation of the Conceptual Design and Mission Profiles of Short-Haul Aircraft," AIAA Paper 86-2696.
- [82] Haberland, C., Fenske, W., Kranz, O., and Stoer, R., 1990, "Computer-Aided Conceptual Aircraft Configuration Development by an Integrated Optimization Approach," ICAS-90-2.6R, *17th Congress of the International Council of the Aeronautical Sciences - ICAS Proceedings*, Stockholm, Sweden, pp. 2164-2173.
- [83] Kroo, I. M., 1992, "An Interactive System for Aircraft Design and Optimization," AIAA Paper 92-1190.
- [84] Raymer, D. P., 1996, "Aircraft Design Optimization on a Personal Computer," AIAA Paper 96-5609.
- [85] Rentema, D. W. E., Jansen, F. W., and Torenbeek, E., 1998, "The Application of AI and Geometric Modelling Techniques in Conceptual Aircraft Design," AIAA Paper 98-4824, *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, pp. 928-935.
- [86] Salas, A. O. and Townsend, J. C., 1998, "Framework Requirements for MDO Application Development," AIAA Paper 98-4740, *7th AIAA/USAF/NASA/ISSMO*

- Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, pp. 261-271.
- [87] Padula, S. L., Korte, J. J., Dunn, H. J., and Salas, A. O., 1999, "Multidisciplinary Optimization Branch Experience Using iSIGHT Software," NASA TM-1999-209714, <http://www.soton.ac.uk/~pbn/MDO/soft.html>
 - [88] Wakayama, S. and Kroo, I. M., 1998, "The Challenge and Promise of Blended-Wing-Body Optimization," AIAA Paper 98-4736.
 - [89] Parkinson, A. R., Balling, R. J., and Free, J. C., 1992, "OptdesX: An X Window-Based Optimal Design Software System," AIAA Paper 92-4759, 4th AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, pp. 559-566.
 - [90] Röhl, P. J., Kolonay, R. M., Irani, R. K., Sobolewski, M., and Kao, K., 2000, "A Federated Intelligent Product Environment," AIAA Paper 2000-4902.
 - [91] Engineous Software, Inc., 2001, "iSIGHT Basic Training Guide Version 6.0."
 - [92] Raymer, D. P., 1999, *Aircraft Design: A Conceptual Approach – Third Edition*, AIAA Publication Series, Reston, VA, Chap. 6.
 - [93] Loftin, L. K., 1980, "Subsonic Aircraft: Evolution and the Matching of Size to Performance," NASA Reference Publication 1060, Hampton, VA.
 - [94] Roskam, J., 1985, *Airplane Design Part I: Preliminary Sizing of Airplanes*, Roskam Aviation and Engineering Corporation, Ottawa, KS, Chap. 3.
 - [95] Teledyne Continental Motors, 2003, "Specification Chart", <http://www.tcmlink.com/producthighlights/ENGTBL.PDF>

- [96] Lycoming, Textron Inc., 2002, "Piston Engine Selection Guide",
<http://www.lycoming.textron.com/main.jsp?bodyPage=productSales/engineSelectionGuide/index.html>
- [97] No Author, 2001-2002, *Jane's All the World's Aircraft*, B.P.C. Publishing.,
London, pp. 831-4.
- [98] Raymer, D. P., 1999, *Aircraft Design: A Conceptual Approach – Third Edition*,
AIAA Publication Series, Reston, VA, pp. 361, Chap. 12.
- [99] Raymer, D. P., 1999, *Aircraft Design: A Conceptual Approach – Third Edition*,
AIAA Publication Series, Reston, VA, pp. 97, Chap. 5.
- [100] Malott, P., 2002, "Performance of the GeoSurv UAV," DR 13-07, Carleton
University, Ottawa, ON.
- [101] Malott, P., 2002, "GeoSurv UAV Drag Polar," DR 23-01, Carleton University,
Ottawa, ON.
- [102] Roskam, J., 1985, *Airplane Design Part I: Preliminary Sizing of Airplanes*,
Roskam Aviation and Engineering Corporation, Ottawa, KS, pp. 119, Chap. 3.
- [103] General Atomics Aeronautical Systems, Inc., 2003, "GNAT System, Remotely
Piloted Aircraft Systems," San Diego, CA.

Appendix A:

Optimization Framework Cost Information (US \$)

As of 1 September 2002

DAKOTA

The DAKOTA software is available free under a GNU general public licence agreement at: <http://endo.sandia.gov/DAKOTA/licensing/license.html>

Epogy

Academic pricing (per seat per number of years) is 20% of the commercial price listed in Table A- 1. If a licence is worth \$5,000¹ or more, four additional licences will be provided at no extra cost on the condition that the licence server would only be used within the Mechanical and Aerospace Engineering department. The maintenance package includes all technical support and upgrades.

Table A- 1: Epogy List Price

Product	Project Complexity	Floating	Floating	Maint.	Fixed	Fixed	Maint.
		10 Year	1 Year	1 Year	10 Year	1 Year	1 Year
Lite(Win/LUX)	P*=25 D**=10					FREE	
Lite(other)	P=25 D=10	\$4,000.00	\$2,000.00	\$600.00	\$2,400.00	\$1,440.00	\$500.00
Level 2	P=50 D=20	\$10,000.00	\$5,000.00	\$1,500.00	\$6,000.00	\$3,600.00	\$900.00
Level 3	P=100 D=30	\$20,000.00	\$10,000.00	\$3,000.00	\$12,000.00	\$7,200.00	\$1,800.00
Level 4	P=300 D=50	\$35,000.00	\$17,500.00	\$5,250.00	\$21,000.00	\$12,600.00	\$3,150.00
Level 5	P=1000 D=200	\$50,000.00	\$25,000.00	\$7,500.00	\$30,000.00	\$18,000.00	\$4,500.00

* Parameters

** Design Variables

iSIGHT

The iSIGHT software is available free with the purchase of a technical support and upgrade package. Table A- 2 summarizes the academic rates based on number of licences required. The annual fees can be negotiated to some extent, in exchange for technical papers on the successful use of iSIGHT that can be used for Engineous Software, Inc. marketing purposes.

¹ All prices are quoted in U.S. Dollars.

Table A- 2: iSIGHT List Price

Number of Licences	Annual Fee
2	\$3,000
3	\$3,900
4	\$4,600
5	\$5,000

LMS OPTIMUS

The academic rate for LMS OPTIMUS is 10% of the commercial rate of \$20,810 (i.e. \$2,081). If an academic project is being funded through an industry source, the rate is 50% of the commercial rate. Additional discounts may apply if a large number of licences are required. Technical support is included in the purchase price.

ModelCenter

The academic rate for ModelCenter is a one-time fee of 20% of the total commercial rate, which may vary from \$13,000 to \$15,000 (~\$3,000). This rate includes 1 year of technical support and software upgrades.

VisualDOC

The academic rate for purchase of VisualDOC is \$1,875, and includes one year of technical support and upgrades. After one year, additional technical support can be purchased at a rate of \$281/year. The academic lease rate for VisualDOC is \$750/year, which includes all technical support and upgrades. All rates include up to 20 floating licences.

Appendix B:
Optimization Framework Vendor Contact Information
As of 1 September 2002

Engineous Software, Inc.
iSIGHT 6.0
Phone: 919-677-6700
Fax: 919-677-8911
2000 CentreGreen Way, Suite 100
Cary, NC 27513
www.engineous.com
info@engineous.com

Jack Cofer, NE USA Sales
Representative and Director of
Aerospace and Turbomachinery
Industries
Phone: 508-376-5307
Fax: 508-376-5385
jack.cofer@engineous.com

LMS International
LMS OPTIMUS Version 3.0
Phone: 248-952-5664
Fax: 248-952-1610
1050 Wilshire Blvd., Suite 250
Troy, MI 48084
www.lmsintl.com
info@lms.be

Les Goldberg, Canadian Sales Rep
Cell: 518-573-6777

Phoenix Integration
ModelCenter and ModelRunner
Phone: 1-800-500-1936 or 540-961-7215
Fax: 540-961-5831
1715 Pratt Drive, Suite 2000
Blacksburg, VA 24060
www.phoenix-int.com
sales@phoenix-int.com

Greg Salow, Sales Rep (Boston)
Phone: 207-799-0067
Cell: 207-329-7216
gsalow@phoenix-int.com

Sandia National Laboratories (SNL)
DAKOTA Version 3.0
P.O Box 5800 Mail Stop 0847
Albuquerque, NM 87185
endo.sandia.gov/DAKOTA/software.html
dakota@sandia.gov

SYNAPS
Epogy and Pointer
Phone: 404-320-7283
Fax: 404-325-1100
2957 Clairmont Road, Suite 170
Atlanta, GA 30329
www.synaps-inc.com/software.html
info@synaps-inc.com
sales@synaps-inc.com

Alex VanderVelden
Phone: 404-320-7283, ext. 222
velden@synaps-inc.com

Vanderplaats Research & Development,
Inc.
DOT 5.0 and VisualDOC 2.0
Phone: 719-473-4611
Fax: 719-473-4638
1767 South 8th Street, Suite 100
Colorado Springs, CO 80906
www.vrand.com
sales@vrand.com

Laura Moreland
Phone: 719-473-4611 ext. 100
laura@vrand.com

Appendix C:

Integrating Matlab as a Simcode into iSIGHT

1. Software Location and Path Definition

The steps for integrating Matlab as a simcode within iSIGHT make several assumptions about the installation location of both programs and the user's "write access" to specific paths.

1.1 iSIGHT Version 7.0

iSIGHT Version 7.0 should be installed in the default directory, C:\Program Files\iSIGHT_70. If the default location was changed during installation, the current location will have to be substituted for the above string in Section 2.3.

To run iSIGHT correctly, the user requires write access to the C:\tmp directory (if full write access to the main hard drive of the computer is not allowed).

1.2 Matlab 5.3 (Release 11)

Matlab 5.3 (Release 11) should be installed in the default directory, C:\MATLABR11. If the default location was changed during installation, the current location will have to be substituted for the above string in Section 2.3.

For Matlab to be able to run user-defined scripts known as M-files, the path to the file must be defined. After starting Matlab, select *Set Path...* under the *File* menu. A new

window will appear with the current path location shown. Add a new path using the *Add to Path...* command under the *Path* menu. Browse for the location of the known M-file and then click *OK*. Close the Path Browser window; in doing so, the program will prompt you to save the changes made to the path for future sessions. Click *Yes* to save the changes; however, the user must have write access to C:\MATLABR11\toolbox\local\pathdef.m to be able to save.

2. Steps to Integrate Matlab as a Simcode Within iSIGHT

To follow the steps presented in the rest of this section, knowledge of typical code-coupling steps within iSIGHT is necessary. Refer to Chapter 4 in [91] for more information.

2.1 Writing the M-file

To integrate Matlab as a simcode in iSIGHT, the analysis must first be written into an M-file. M-files are a series of commands or computations stored in ASCII text. Figure C- 1 is an example M-file, used to compute a Mach number from given values of altitude and velocity. The procedure for calculating Mach number from altitude and velocity in MachNumber.m was adapted from the public access FORTRAN code.¹ For ease of illustration, this example will be carried through the remainder of this section.

¹ Carmichael, R. L., 2002, "Definition of the 1976 Standard Atmosphere," Public Domain Aeronautical Software (PDAS), <http://www.pdas.com/coesa.htm>

Figure C- 1: MachNumber.m

```
% MOCA M-File
% SetInitialValues
% Calculate Mach Number at a Given Altitude in FPS Units of Standard
% Atmosphere

% Altitude in Feet
Altitude=150.0;
% Convert to Geopotential Altitude
R_earth=6369.0*3280.83;
h=Altitude*R_earth/(Altitude+R_earth);

% Input Velocity in Knots
V=100.0;
% Convert to Feet/Second
Vfps=V*1.68781;

h_initial=[0.0; 11.0; 20.0; 32.0; 47.0; 51.0; 71.0; 84.852]*3280.83;
dT_dh=[-6.5; 0.0; 1.0; 2.8; 0.0; -2.8; -2.0; 0.0]*1.8/3280.83;
T_initial=[288.15; 216.65; 216.65; 228.65; 270.65; 270.65; 214.65;
186.946]*1.8;

% Compute Temperature Based on Altitude to a Maximum of ~ 282,000 ft
(86 Km)
% Using a Binary Search
i=1;
j=8;
l=0;
while l==0
    k=floor((i+j)/2);
    if h < h_initial(k)
        j=k;
    else
        i=k;
    end
    if j <= i+1
        l=1;
    else
        l=0;
    end
end
T=T_initial(i)+dT_dh(i)*(h-h_initial(i));

% Compute Mach Number
Ma=Vfps/sqrt(1.40*1717*T);

% Output to File
filename = 'Mcruiase.txt';
file = strcat('C:\iSIGHT\GeoSurvUAV\',filename);
fid = fopen(file,'w+t');
fprintf(fid,'Mach Number = %0.5f',Ma);
fclose(fid);

exit
```

2.2 Creating Input and Output Files

Once the M-file has been written, with all input and output parameters defined, any required input and output files must be created. There are several different ways to define input and output files, depending on the structure of the M-file. The following procedure is recommended as a robust method for calling a single M-file repeatedly.

M-files can be either scripts or functions. Scripts are simply files containing a sequence of Matlab statements. Functions make use of their own local variables and accept variable input arguments.² As a result, the M-file can be easily re-used within Matlab and called by other M-files. Figure C- 2 illustrates how the previous MachNumber.m file was modified from a script to a function. Instead of setting the values of altitude and velocity, the function requires them as an input.

The function M-file can be called by another M-file, which iSIGHT can parse as an input file. Figure C- 3 shows an M-file that calls MachNumber.m using variable input values, and writes the returned value of Mach number to an output text file that iSIGHT can parse. Figure C- 4 displays the results. While number of files could be reduced by having Matlab write to a single script file, it is poor code practice and can generate errors if the program tries to read and write to a single file at the same time.

² The MathWorks, Inc., 2002, "Matlab Function Reference,"
<http://www.mathworks.com/access/helpdesk/help/techdoc/ref/ref.shtml>

Figure C- 2: MachNumber.m (Modified)

```
% MOCA Input File
% SetInitialValues
% Calculate Mach Number at a Given Altitude in FPS Units of Standard
% Atmosphere

function Ma = MachNumber(Altitude,V)

% Convert Altitude to Geopotential Altitude
R_earth=6369.0*3280.83;
h=Altitude*R_earth/(Altitude+R_earth);

% Convert Velocity to Feet/Second
Vfps=V*1.68781;

h_initial=[0.0; 11.0; 20.0; 32.0; 47.0; 51.0; 71.0; 84.852]*3280.83;
dT_dh=[-6.5; 0.0; 1.0; 2.8; 0.0; -2.8; -2.0; 0.0]*1.8/3280.83;
T_initial=[288.15; 216.65; 216.65; 228.65; 270.65; 270.65; 214.65;
186.946]*1.8;

% Compute Temperature Based on Altitude to a Maximum of ~ 282,000 ft
(86 Km)
% Using a Binary Search
i=1;
j=8;
l=0;
while l==0
    k=floor((i+j)/2);
    if h < h_initial(k)
        j=k;
    else
        i=k;
    end
    if j <= i+1
        l=1;
    else
        l=0;
    end
end
T=T_initial(i)+dT_dh(i)*(h-h_initial(i));

% Compute Mach Number
Ma=Vfps/sqrt(1.40*1717*T)
```

Figure C- 3: MachIO.m

```
% MOCA Input File
% SetInitialValues
% Input File to MachNumber.m

% Altitude in Feet
Altitude=150.0;

% Input Velocity in Knots
V=100.0;

Ma = MachNumber(Altitude,V);

% Output to File
filename = 'MachNumber.txt';
file = strcat('C:\iSIGHT\MFiles\ ',filename);
fid = fopen(file,'w+t');
fprintf(fid,'Mach Number = %0.5f',Ma);
fclose(fid);

exit
```

Note the use of the *exit* command to close Matlab after the M-file has run. This step is necessary to avoid multiple window generation each time iSIGHT executes the simcode call.

Figure C- 4: MachNumber.txt

Mach Number = 0.15124

2.3 Setting up the Simcode Block

An M-file can be added as a simcode block to iSIGHT after the input and output files have been completely defined. The procedure for inserting a Simcode into a task is found in [91]; only the modifications to the procedure are presented here.

Subsequent to adding a simcode block to a task in the Process Integration window and renaming the block, the properties of the input file should be defined as shown in Figure C- 5. In this case, MachIO.m is specified as both the Template File and Input File. A separate template file could be copied from input file and used if the file is large and iSIGHT need only access a small portion of the file.

Figure C- 5: Input Properties Window

The instructions for parsing the input file can be defined after the input file name is specified, and would be tailored to the input required by the M-file. Figure C- 6 displays the input file parsing instructions for MachIO.m. It is important to note the additional step at the end of each input variable replacement. A semi-colon is directly written to the file to stop the display of the value of the variable on the Matlab screen. The additional

character cleans up the M-file output and reduces the processing time required by the analysis (i.e. more time is required to display characters than to process the entire function).

Figure C- 6: Input File Parse Window

Next, the simcode program properties are specified. To eliminate the appearance of the command line window with every call to Matlab, conspaw.exe (a program provided by iSIGHT for this purpose, located in the directory shown in Figure C- 7) is called as the Simulation Code. Matlab is called as an Optional Argument in the command line, followed by a command to open the Matlab window minimized. The remaining argument

is a command to Matlab to run the input M-file, since MachIO.m calls MachNumber.m. Observe the static nature of the call to the M-file in Figure C- 7; it is because of this that MachNumber.m is not called directly from iSIGHT. Constant input values could be included in the optional argument line (e.g. `-r MachNumber(10000,400)`), but the M-file would not be able to receive variable input values.

Figure C- 7: Simcode Program Properties

Before closing the Program Properties window, a default setting on the Run Control tab must be changed. Check the box labelled “Restore original user environment prior to execution,” as shown in Figure C- 8. This setting is required to run Matlab correctly outside of the iSIGHT shell environment. No other default settings need to be changed.

Figure C- 8: Simcode Run Control

The output file properties, shown in Figure C- 9, are straightforward – only the text file written by MachIO.m is specified.

Figure C- 9: Output Properties Window

The output file parsing instructions are unchanged from the standard procedure; Figure C- 10 is shown only for reference.

Figure C- 10: Output File Parse Window

When all of the simcode properties are set correctly, the Process Integration window should look similar to Figure C- 11. In the task pane to the left of the window, two additional tasks, entitled Sleep1 and Sleep2, can be seen. The addition of these Tcl scripts will be discussed in the next section.

Figure C- 11: Simcode Process

2.4 Adding Tcl Scripts

A Tcl script must be added in sequence, both before and after, each simcode block that calls Matlab. The script forces iSIGHT to halt processing for a specified period of time, which provides Matlab the time necessary to open, access the M-files called in the simcode block, write the output, and close. If the sleep script is not added, iSIGHT attempts to access the files while Matlab is still reading and writing to them. As a result, data from a previous design point is read into the current point, misleading the optimization algorithm and storing incorrect information in the database.

To add a Tcl script, select *Tcl Script* under the *Insert* menu (no button exists for this procedure). The Tcl Script Editor window will appear, as in Figure C- 12. Enter the command *exp_sleep x*, where *x* is the time value in seconds. The sleep time may need to be modified, depending on processor speed and the number of programs running in the background system. The minimum sleep time acceptable for a PIII 1GHz with 256 Mb of RAM running Windows XP Professional was found to be 0.4 seconds.

Figure C- 12: Tcl Script Editor Window

Select *OK* and repeat the process. The Process Integration window should now resemble Figure C- 13. The tasks as shown will execute in parallel; however, sequential execution is necessary. Section 2.5 explains how to sequence the tasks.

Figure C- 13: Unsequenced Tasks

2.5 Sequencing Tasks

iSIGHT automatically sequences tasks for execution based on the input and output data defined for each task. If two tasks do not share data, they can be executed in parallel (i.e. at the same time). This decreases the run time of the overall process. The purpose of adding the Tcl scripts in the previous section is to stop iSIGHT from executing too quickly for Matlab to run correctly. Thus, a particular sequence of tasks set by the user is desired.

To sequence the sleep and simcode tasks, select *Sequence* from the *Edit* menu. The Sequence window, as illustrated in Figure C- 14, will appear. Select a Tcl sleep script block and click on the *Add* button. Repeat in order for the simcode block and the remaining Tcl block and click *OK*.

Figure C- 14: Sequence Window

The Process Integration window should now appear similar to Figure C- 15. The blocks are now in sequential order and the entire task should execute correctly.

Figure C- 15: Sequenced Tasks

3. Conclusion

The steps required to integrate Matlab properly as a simcode were presented, with reference to the standard steps for coupling simulation codes to iSIGHT. While these steps are applicable only to iSIGHT Version 7.0, any version of Matlab could be coupled using the same method, provided changes to the path name of the executable file are taken into account.

Appendix D:

Integrating Pro/Engineer as a Simcode into iSIGHT

1. Software Location and Path Definition

The steps for integrating Pro/Engineer as a simcode within iSIGHT make several assumptions about the installation location of both programs and the user's "write access" to specific paths.

1.1 iSIGHT Version 7.0

iSIGHT Version 7.0 should be installed in the default directory, C:\Program Files\iSIGHT_70. If the default location was changed during installation, the current location will have to be substituted for the above string in Section 2.2.

To run iSIGHT correctly, the user requires write access to the C:\tmp directory (if full write access to the main hard drive of the computer is not allowed).

1.2 Pro/Engineer 2001

Pro/Engineer 2001 should be installed in the default directory, C:\PTC. If the default location was changed during installation, the current location will have to be substituted for the above string in Section 2.2.

2. Steps to Integrate Pro/Engineer as a Simcode Within iSIGHT

To follow the steps presented in the rest of this section, knowledge of typical code-coupling steps within iSIGHT is necessary. Refer to Chapter 4 in [91] for more information.

2.1 Writing the Trail File

To integrate Pro/Engineer as a simcode in iSIGHT, the 3-D solid model information must first be written into a “trail file”. A trail file is an ASCII text file that logs the steps taken by a user in a single Pro/Engineer session. The trail file can be “replayed” later, executing the same steps the user performed automatically. The trail file, initially named “trail.txt.#” (following the standard Pro/Engineer number scheme), must be renamed to be able to replay it. The trail file “moca.txt,” as shown in Figure D- 1, will be used as an example throughout the remaining steps.

Figure D- 1: Moca.txt

```
!trail file version No. 1151
!Pro/ENGINEER TM 2001 (c) 1988-2001 by Parametric Technology Corporation All Rights Reserved.
~ Select `main_dlg_cur` `MenuBar1` \
1 `File`
~ Close `main_dlg_cur` `MenuBar1`
~ Activate `main_dlg_cur` `File.psh_working_dir`
~ Select `file_open` `Ph_path.Path` \
1 `C:`
~ Select `file_open` `Ph_list.Filelist` \
1 `iSIGHT`
~ Activate `file_open` `Ph_list.Filelist` \
1 `iSIGHT`
~ Select `file_open` `Ph_list.Filelist` \
1 `ProE`
```

```

~ Activate `file_open` `Ph_list.Filelist` \
1 `ProE`
~ Activate `file_open` `Open`
!%CISuccessfully changed to C:\iSIGHT\ProE\ directory.
~ Activate `main_dlg_cur` `ProCmdModelNew.file`
~ Input `new` `InputPanel1` \
`fuselage1`
~ Update `new` `InputPanel1` \
`fuselage1`
~ Activate `new` `chk_use_default_template` \
0
~ Activate `new` `OK`
!Command ProCmdModelNewExe was pushed from the software.
~ FocusIn `new_file_opts` `psh_ok`
~ Activate `new_file_opts` `psh_browse`
~ FocusOut `new_file_opts` `new_file_opts`
~ Select `file_open` `Ph_list.Filelist` \
1 `fps_part_solid.prt`
~ Activate `file_open` `Open`
~ FocusIn `new_file_opts` `psh_browse`
~ Activate `new_file_opts` `psh_ok`
!31-Jun-03 12:00:00 Start C:\iSIGHT\ProE\fps_part_solid.prt.1
!31-Jun-03 12:00:03 End C:\iSIGHT\ProE\fps_part_solid.prt.1
!31-Jun-03 12:00:04 Start C:\iSIGHT\ProE\fuselage1.prt.1
!31-Jun-03 12:00:04 End C:\iSIGHT\ProE\fuselage1.prt.1
#FEATURE
#CREATE
#PROTRUSION
#ADVANCED
#DONE
#FROM FILE
#DONE
!%CPSpecify coordinate system that file entities are with respect to.
!%CPSelect coordinate system from menu.
#PRT_CSYS_DEF
!%CPEnter imported blend file name [quit]:
!%TR
fuselage1
!%CPArrow points TOWARD area to be added. Pick FLIP or OKAY.
#OKAY
!%CPAll elements have been defined. Select element(s) or action(s) from dialog box.
~ Activate `Odui_Dlg_01` `okay`
!%CPAll elements have been defined. Select element(s) or action(s) from dialog box.
~ Activate `main_dlg_cur` `ProCmdModelSave.file`
!%CPEnter object to save [fuselage1.PRT]
!%TR

```

2.2 Setting up the Simcode Block

The procedure for inserting a Simcode into a task is found in [91]; only the modifications to the procedure are presented here. Subsequent to adding a simcode block to a task in the Process Integration window and renaming the block, the properties of the input and output files would ordinarily have to be defined. In the case of the “moca.txt” example, no input or output files are required. As a result, both the Input and Output blocks in the Process Integration window can be deleted. Only the simcode block remains, as shown in Figure D- 2. If input and/or output files are required, the standard procedure found in [91] should be followed.

Figure D- 2: Process Integration Window

Next, the simcode program properties are specified. Pro/Engineer is called as the Simulation Code as shown in Figure D- 3. The trail file, “moca.txt,” is called as an Optional Argument in the command line.

Figure D- 3: Simcode Program Properties

3. Conclusion

The steps required to integrate Pro/Engineer properly as a simcode were presented, with reference to the standard steps for coupling simulation codes to iSIGHT. While these steps are applicable only to iSIGHT Version 7.0, any version of Pro/Engineer could be coupled using the same method, provided changes to the path name of the executable file are taken into account.

Appendix E:
Engine and Boom Placement

Figure E- 1: Propeller Engine Placement, One Engine

Figure E- 2: Propeller Engine Placement, Two Engines

Figure E- 3: Propeller Engine Placement, Three Engines

Figure E- 4: Propeller Engine Placement, Four Engines

Figure E- 5: Jet Engine Placement, One Engine

Figure E- 6: Jet Engine Placement, Two Engines

Figure E- 7: Jet Engine Placement, Three Engines

Figure E- 8: Jet Engine Placement, Four Engines

Figure E- 9: Boom Placement