HIGH-FIDELITY MULTIDISCIPLINARY DESIGN OPTIMIZATION OF FLEXIBLE AIRCRAFT WINGS

by

Brian T. Leonard
B.Eng, Ryerson University, Toronto, 2004

A THESIS
PRESENTED TO RYERSON UNIVERSITY
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

IN THE PROGRAM OF

MECHANICAL ENGINEERING

TORONTO, ONTARIO, CANADA, 2006

Author's Declaration

I hereby declare that I am the sole author of this thesis. I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

Brian Leonard

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Brian Leonard

High-Fidelity Multidisciplinary Design Optimization of Flexible Aircraft Wings

Master of Applied Science
In the Program of
Mechanical Engineering
2006

Brian T. Leonard

School of Graduate Studies
Ryerson University

Abstract

Multidisciplinary design optimization (MDO) was performed on an aircraft wing using high-fidelity design tools. The wing aerodynamics were analyzed using computational fluid dynamics (CFD) with FLUENT and the wing structure was analyzed via finite element analysis (FEA) in ANSYS. MATLAB was used as a wrapper to perform computational static aeroelastic analysis on any wing configuration using the aforementioned high-fidelity tools. A main program was developed to convert pressures to forces, map the CFD grid to the FEA mesh, and to transfer the FEA displacements back to the CFD grid. The static aeroelastic software was coupled with the multidisciplinary design feasible (MDF) MDO architecture using sequential quadratic programming (SQP) to perform the optimization. The optimization was given the maximum amount of design freedom to create any wing shape. Ultimately, it was found that MDO is possible using these high-fidelity tools and that, to get a true wing design, aeroelastic effects must be included in the MDO procedure.

Acknowledgements

I would first like to thank my supervisor Dr. Kamran Behdinan for his support during the process of completing this thesis. I would also like to thank him for giving me many opportunities during my tenure at Ryerson that have allowed me to become a better engineer and researcher.

Secondly, I would like to thank my co-supervisor Dr. Zouheir Fawaz for seeing potential in me that I did not even know that I had. His guidance over the years has allowed me to gain the confidence to get me to where I am today.

I would also like to thank Mr. Ruben Perez for his help in all aspects of MDO and aircraft design. Without his help regarding MDO architectures, this thesis would never have been completed in time. His enthusiasm toward this research was infectious and it helped me get through the tough times.

Table of Contents

AUTHOR'S DECLARATION iii

ABSTRACT v

ACKNOWLEDGEMENTS vii

LIST OF FIGURES xiii

LIST OF TABLES xvii

NOMENCLATURE xix

CHAPTER 1. INTRODUCTION 1

1.1 FUTURE OF DESIGN 1

1.2 WING OPTIMIZATION PROBLEM 2

1.3 THESIS LAYOUT 3

CHAPTER 2. MULTIDISCIPLINARY DESIGN OPTIMIZATION (MDO) 5

2.1 MDO METHODOLOGIES 5

2.1.1 Multi-Disciplinary Feasible 6

2.1.2 Individual Discipline Feasible 8

2.1.3 Collaborative Optimization 9

2.1.4 Concurrent Sub Space Optimization 12

2.1.5 Bi-Level Integrated System Synthesis 13

2.2 MDO IN AIRCRAFT DESIGN 15

2.3 OPTIMIZATION METHODS 17

2.3.1 Sequential Linear Programming *17*
2.3.1.1 Algorithm Development 18
2.3.2 Sequential Quadratic Programming *19*
2.3.2.1 Algorithm Development 21

CHAPTER 3. HIGH-FIDELITY AIRCRAFT OPTIMIZATION 25

3.1 CURRENT RESEARCH 25
3.1.1 Aeroelastic Coupling *29*
3.1.2 CFD/FEM Mesh Mapping *31*

CHAPTER 4. AERO-STRUCTURAL ANALYSIS 33

4.1 MODEL GEOMETRY 33
4.1.1 NACA 4-Digit Airfoil *33*
4.1.2 Parametric Aerodynamic Wing Shape *35*
4.1.3 Consistent Wing Geometry *37*
4.2 COMPUTATIONAL FLUID DYNAMICS (CFD) 38
4.2.1 Grid Generation *38*
4.2.1.1 Grid Skewness 40
4.2.1.2 Grid Continuity 41
4.3 FINITE ELEMENT METHOD (FEM) 41
4.3.1 Model Creation *42*
4.3.2 Mesh Generation *43*
4.3.3 Structural Taper *46*
4.4 AERO-STRUCTURAL COUPLING 46
4.4.1 Load Transfer *48*
4.4.2 Displacement Transfer *51*

CHAPTER 5. AERO-STRUCTURAL OPTIMIZATION RESULTS 53

5.1 SENSITIVITY ANALYSIS 53
5.2 CFD VALIDATION 54
5.2.1 2D NACA Airfoil *54*
5.2.2 Onera M6 Wing *56*

5.3 STATIC AEROELASTIC CONVERGENCE STUDY 57
5.4 OPTIMIZATION SETUP 60
5.4.1 Objective Function *61*
5.4.2 Constraints *61*
5.5 WING OPTIMIZATION RESULTS 63
5.5.1 Rigid Wing Results *63*
5.5.2 Flexible Wing Results *69*

CHAPTER 6. CONCLUSION AND FUTURE WORK 77

REFERENCES 81

APPENDIX A. MDO NUMERICAL EXAMPLES 89

A.1 MDF NUMERICAL EXAMPLE 90
A.2 IDF NUMERICAL EXAMPLE 91
A.3 CO NUMERICAL EXAMPLE 92

APPENDIX B. MATLAB WRAPPER 97

B.1 GAMBIT MODULE 98
B.1.1 Interfacing with GAMBIT *98*
B.1.2 Journal Code Example Command *98*
B.1.3 Calling GAMBIT from MATLAB *99*
B.2 FLUENT MODULE 100
B.2.1 Interfacing with FLUENT *100*
B.2.2 Journal Code Example Command *100*
B.2.3 Calling FLUENT from MATLAB *101*
B.3 ANSYS MODULE 102
B.3.1 Interfacing with ANSYS *102*
B.3.2 APDL Example Command *103*
B.3.3 Calling ANSYS from MATLAB *103*
B.4 MATLAB WRAPPER DATA FLOW 104
B.5 MATLAB WRAPPER FILE DESCRIPTIONS 111

APPENDIX C. EXAMPLE ANSYS APDL CODE 115

List of Figures

FIGURE 1-1: THE DIFFERENT LEVELS OF DESIGN DETAIL 3

FIGURE 2-1: MULTI-DISCIPLINARY FEASIBLE ARCHITECTURE [23] 7
FIGURE 2-2: INDIVIDUAL DISCIPLINE FEASIBLE ARCHITECTURE [23] 8
FIGURE 2-3: COLLABORATIVE OPTIMIZATION ARCHITECTURE [23]. 10
FIGURE 2-4: CONCURRENT SUB SPACE OPTIMIZATION ARCHITECTURE [23] 12
FIGURE 2-5: BI-LEVEL INTEGRATED SYSTEM SYNTHESIS ARCHITECTURE [23]. 14
FIGURE 2-6: THE MANY DIFFERENT DISCIPLINES IN AIRCRAFT DESIGN [23] 16
FIGURE 2-7: VISUALIZATION OF SLP 19
FIGURE 2-8: FLOW CHART REPRESENTATION OF SLP 20
FIGURE 2-9: FLOW CHART REPRESENTATION OF SQP 23

FIGURE 3-1: THE CFD/CSM COUPLING PROCEDURE [47] 30
FIGURE 3-2: THE FINITE INTERPOLATION ELEMENT [58] 32

FIGURE 4-1: TYPICAL AIRFOIL NOMENCLATURE 34
FIGURE 4-2: DEFINITION OF NACA 4-DIGIT MEAN CAMBER LINE 35
FIGURE 4-3: WING MODEL PARAMETRIC DEFINITIONS 36
FIGURE 4-4: CFD FAR-FIELD C-GRID 38
FIGURE 4-5: EXAMPLE OF WING CFD MODEL 39

FIGURE 4-6: EXAMPLE OF NUMERICAL NOISE 40
FIGURE 4-7: EXAMPLE OF GRID SKEWNESS 40
FIGURE 4-8: EXAMPLE OF GRID CONTINUITY 41
FIGURE 4-9: STRUCTURAL ELEMENTS WITHIN THE WING 43
FIGURE 4-10: LOCATION OF A *HOT SPOT* FOR AN AREA 44
FIGURE 4-11: FINITE ELEMENTS USED TO MODEL WING 45
FIGURE 4-12: FEM MESH OF THE STRUCTURAL ELEMENTS 45
FIGURE 4-13: STATIC AEROELASTIC COUPLING PROCEDURE 46
FIGURE 4-14: PERCENT ERROR OF CALCULATED LIFT AND DRAG VERSUS FLUENT LIFT AND DRAG 48
FIGURE 4-15: CFD GRID AREA CALCULATION 49
FIGURE 4-16: MAPPING CFD GRID POINTS TO FEM NODES 51

FIGURE 5-1: ONERA M6 WING GEOMETRY 56
FIGURE 5-2: ONERA M6 WING VALIDATION 57
FIGURE 5-3: AEROELASTIC DEFLECTION OF AN ARBITRARY WING CONFIGURATION 58
FIGURE 5-4: STATIC AEROELASTIC CONVERGENCE STUDY RESULTS 59
FIGURE 5-5: ORIGINAL AND OPTIMIZED RIGID WING SHAPE 64
FIGURE 5-6: RIGID WING CONVERGENCE OF THE OPTIMIZATION OBJECTIVE FUNCTION 65
FIGURE 5-7: PRESSURE DISTRIBUTION OF THE ORIGINAL AND OPTIMIZED WING SHAPES 66
FIGURE 5-8: RIGID WING CONVERGENCE OF LIFT EQUALS WEIGHT CONSTRAINT 67
FIGURE 5-9: RIGID WING CONVERGENCE OF LIFT-TO-DRAG RATIO 67
FIGURE 5-10: RIGID WING CONVERGENCE OF STRESS CONSTRAINT 69
FIGURE 5-11: RIGID WING CONVERGENCE OF TOTAL WING WEIGHT 69
FIGURE 5-12: ORIGINAL, OPTIMIZED RIGID, AND OPTIMIZED FLEXIBLE WING SHAPES 70
FIGURE 5-13: FLEXIBLE WING CONVERGENCE OF THE OPTIMIZATION OBJECTIVE FUNCTION 71
FIGURE 5-14: FLEXIBLE WING CONVERGENCE OF LIFT EQUALS WEIGHT CONSTRAINT 72
FIGURE 5-15: PRESSURE DISTRIBUTION OF THE OPTIMIZED RIGID AND OPTIMIZED FLEXIBLE WING SHAPES 73
FIGURE 5-16: FLEXIBLE WING CONVERGENCE OF LIFT-TO-DRAG RATIO 74

FIGURE 5-17: FLEXIBLE WING CONVERGENCE OF STRESS CONSTRAINT 75
FIGURE 5-18: FLEXIBLE WING CONVERGENCE OF TOTAL WING WEIGHT 75

FIGURE A-1: EXAMPLE OF CO PROBLEM FORMULATION 93
FIGURE A-2: A. CONVERGENCE OF NUMERICAL EXAMPLE. B. ZOOMED CONVERGENCE 94

FIGURE B-1: SYMBOL IDENTIFICATION FOR MATLAB WRAPPER FLOW CHART 105
FIGURE B-2: COLOUR IDENTIFICATION FOR MATLAB WRAPPER FLOW CHART 105

List of Tables

TABLE 5-1: 2D AIRFOIL BENCHMARKING RESULTS 55
TABLE 5-2: AERODYNAMIC CONFIGURATION FOR STATIC-AEROELASTIC TEST CASES 57
TABLE 5-3: STRUCTURAL CONFIGURATION FOR STATIC-AEROELASTIC TEST CASES 58
TABLE 5-4: AERODYNAMIC DESIGN VARIABLE BOUNDS 62
TABLE 5-5: STRUCTURAL DESIGN VARIABLE BOUNDS 62
TABLE 5-6: STRUCTURAL MATERIAL PROPERTIES OF ALUMINUM 62
TABLE 5-7: RIGID WING DESIGN VARIABLE BOUNDS, INITIAL AND OPTIMUM POINTS 65
TABLE 5-8: FLEXIBLE WING DESIGN VARIABLE BOUNDS, INITIAL AND OPTIMUM POINTS ... 71

Nomenclature

AIAA	-	American Institute of Aeronautics and Astronautics
APDL	-	ANSYS Parametric Design Language
BFGS	-	Broydon-Fletcher-Goldfarb-Shanno
BLISS	-	Bi-Level Integrated System Synthesis
CFD	-	Computational Fluid Dynamics
CO	-	Collaborative Optimization
CSM	-	Computational Solid Mechanics
CSSO	-	Concurrent Sub Space Optimization
FE	-	Finite Element
FEA	-	Finite Element Analysis
FEM	-	Finite Element Method
FSI	-	Fluid Structure Interaction
GUI	-	Graphical User Interface
GSE	-	Global Sensitivity Equation
IDF	-	Individual Discipline Feasible
KKT	-	Karush-Kuhn-Tucker
LP	-	Linear Programming
MDA	-	Multidisciplinary Analysis
MDF	-	Multidisciplinary Design Feasible
MDO	-	Multidisciplinary Design Optimization
MPC	-	Multi-Point Constraint

NURBS	-	Non-Uniform Rational B-Splines
QP	-	Quadratic Programming
SAADO	-	Simultaneous Aerodynamic Analysis and Design Optimization
SFC	-	Specific Fuel Consumption
SLP	-	Sequential Linear Programming
SQP	-	Sequential Quadratic Programming
WingMOD	-	Wing Multidisciplinary Optimization Design

Chapter 1

Introduction

1.1 Future of Design

With the advent of more and more complex engineering systems, balancing hundreds of design variables is an impossible task for a design team. The future of engineering will require a delicate balance between all design variables to find the optimal design point. In order to find this balance an optimization methodology known as multidisciplinary design optimization (MDO) will be the future of design. MDO can be used for any system where more than one discipline is used during the design. For example, three closely related disciplines in aerospace design are structural, aerodynamic, and control systems. Using MDO all three disciplines can be optimized to meet specific objective functions and obtain a delicate balance. The use of MDO will become a standard within all industries because of the need to decrease the design and redesign costs, as well as remain competitive in the global market.

The research that was done for this thesis is high-fidelity multidisciplinary design optimization of aircraft wings. The term high-fidelity refers to the use of modern engineering design tools such as computational fluid dynamics (CFD) and the finite element method (FEM) in order to obtain highly sophisticated models of the problem. High-fidelity MDO as a research topic has been gaining popularity over the past few years, primarily due to the increase in the computational solving times of the high-fidelity

design tools. This type of optimization is still in its infancy and many areas of research still remain untouched. The research that is contained in this thesis is the optimization of an aircraft wing while considering both the structures discipline and the aerodynamics discipline at the same time. This is important because several researchers have shown that a large increase in the overall aircraft performance can be obtained when both of the above disciplines are considered simultaneously [1].

1.2 Wing Optimization Problem

During the design process several different specialists work together in an iterative process in order to find an optimal design. This type of process can take a long time because of the need to design and redesign components. For example, the aerodynamics designer may want to increase the sweep of the wing to help decrease the effect of the shockwave on the wing [2, 3] and thus decrease drag. The controls designer may want to decrease the wing sweep because it may decrease the maneuverability of the aircraft because of the increased pitching moment. Finally, the structures designer may also want to decrease sweep because the amount of lift, drag, or pitching moment will change the wing stress. Overall, it requires each discipline designer to compromise in order to find which sweep configuration will satisfy each designers constraints while having the ideal performance. This type of process is rather simple when only one design variable is being considered. However, when many design variables are being used this process can a take a long time and it is not guaranteed that the best system configuration will be found.

The purpose of this research is to create an automated optimization routine that can use high-fidelity commercial software packages in order to find the best wing configuration when considering both the aerodynamics and structures disciplines. The overall optimization procedure is done using a sophisticated sequential quadratic programming (SQP) routine that is available in MATLAB [4]. The optimization routine was modified from the original MATLAB source code so that the optimal could be found quicker when compared against the original MATLAB optimization routine. The overall coupling of the two disciplines is also done using MATLAB. The aerodynamics discipline is controlled using GAMBIT [5] for geometry creation and grid creation, while

the solving is done using FLUENT [6]. The structures discipline is controlled by ANSYS [7]. The aerodynamics discipline has 5 design variables that change the wing taper, semi-span, sweep, root chord, and thickness/chord ratio. The structures discipline has 7 design variables that can change the wing upper skin thickness, lower skin thickness, spar thickness, rib thickness, stringer height, stringer width, and finally the structural taper ratio. Using such high-fidelity design tools as the ones mentioned above results in a solution that provides a good estimate of the preliminary design. Preliminary design is considered to be a moderate level of design detail that sits between conceptual design and detailed design as seen in Figure 1-1 [2]. The downside to using these tools is that the time needed to achieve convergence for the optimization is higher than analytical methods.

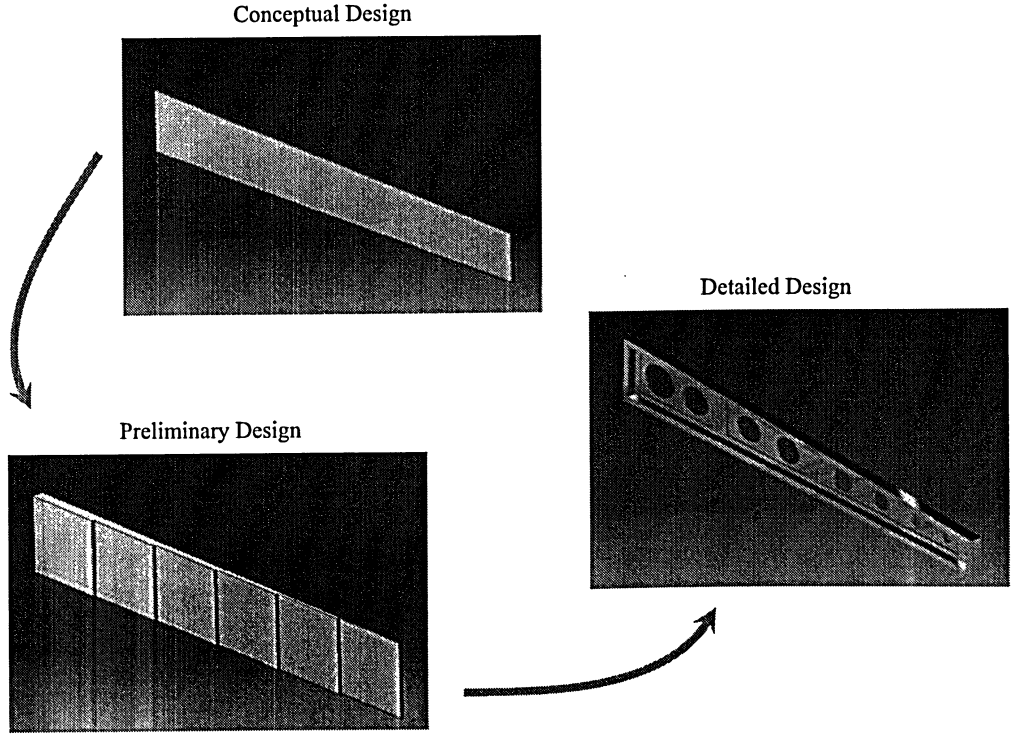


Figure 1-1: The different levels of design detail

1.3 Thesis Layout

The purpose of this thesis is to give any reader that is well versed in optimization, aerodynamics, aero-structures, and design the ability to continue this research. The second chapter provides an introduction to MDO techniques. Each of the main MDO

methods is discussed and how these are used in current aircraft design research. In addition, two sequential programming techniques are introduced to display the computational power required for this type of optimization. The third chapter contains detailed information about the current state of high-fidelity aircraft optimization. This includes variable fidelity methods, aeroelastic coupling methods, and CFD/FEM mesh mapping techniques. Chapter four gives details how the wing model geometry is defined and created. Information concerning the CFD and FEM model generation and CFD/FEM aero-structural coupling are included. The fifth chapter provides the results of the sensitivity analysis performed for the optimization and validation test cases for the CFD. The methodologies used in the optimization setup are included, along with the optimization results. Chapter six details the conclusions and the overall results of this research. More importantly it provides detailed recommendations for improvements that need to be made for any one who wishes to continue on with this work.

Chapter 2

Multidisciplinary Design Optimization (MDO)

This chapter will discuss several MDO methods, how to choose which method to use for a specific problem, and aerospace design problems that use MDO. In addition, sequential programming optimization techniques are introduced.

2.1 MDO Methodologies

To provide a background for this work an introduction of several MDO methods are discussed here. MDO is a methodology used in the design of systems where there is the interaction of many different disciplines present. For example, Sobieski [8] has given examples of aircraft optimization where structural, aerodynamic, and control disciplines are optimized at the same time. This method of optimization helps to answer the question presented by the American Institute of Aeronautics and Astronautics (AIAA) [9] "How to decide what to change, and to what extent to change it, when everything influences everything else." Integrating many disciplines together during optimization may help to answer this question for designers. The methods of optimization discussed here can be extended to include nearly any multidisciplinary design.

In general an MDO problem can be broken down into two major categories [10], the formulation architecture of the problem and the selection of the optimization routine. The formulation architecture of the problem is necessary to turn the multidisciplinary design in to a mathematical statement that can be used in the optimization. The

architecture is an inseparable part of the problem formulation and thus is the single most important aspect of MDO. Depending on the problem at hand the proper choice of architecture can help solve the problem quickly and accurately. The selection of the optimization routine is also extremely important because different optimization algorithms will work better in certain situations and may not work in others. The optimization algorithm depends on the situation; some researchers have used gradient based numerical optimization, genetic algorithms [11], or non-linear programming models [9] just to name a few. The proper choice of optimization routine is up to the discretion of the designer.

As mentioned previously the architecture is a very important aspect of MDO. There are several methods that have been developed and can be used to solve an MDO problem. The five main architectures for MDO are, Multi-Disciplinary Feasible (MDF) [12,13,14,15], Individual Discipline Feasible (IDF) [11,12,14], Collaborative Optimization (CO) [10,16,17,18,19], Concurrent Sub Space Optimization (CSSO), and Bi-Level Integrated System Synthesis (BLISS) [20,21,22].

2.1.1 Multi-Disciplinary Feasible

Multi-Disciplinary Feasible (MDF) method is the most common way of solving an MDO problem and it is the easiest to implement. The MDF method works by achieving multidisciplinary feasibility upon every iteration of the optimization. Multidisciplinary feasibility essentially means that upon each iteration all of the design variables satisfy each discipline and there are no conflicting variables. The entire architecture of the MDF method is shown in Figure 2-1. For simplicity Figure 2-1 shows only 3 disciplines, but it can be extended for N number of disciplines. The architecture links a single optimizer with a multidisciplinary design analysis (MDA). The optimizer is responsible for selecting both global (z) and local (x) design variables for the MDA. Achieving a multidisciplinary feasible design essentially provides the coupling (y) variables to the system. This is usually done by performing Gauss-Seidel (fixed-point iteration) procedure for coupled non-linear equations [14] upon each iteration of the optimization. Once the feasibility is achieved all design variables are known and the analysis calculates the values of the objective function and the constraints for use in the

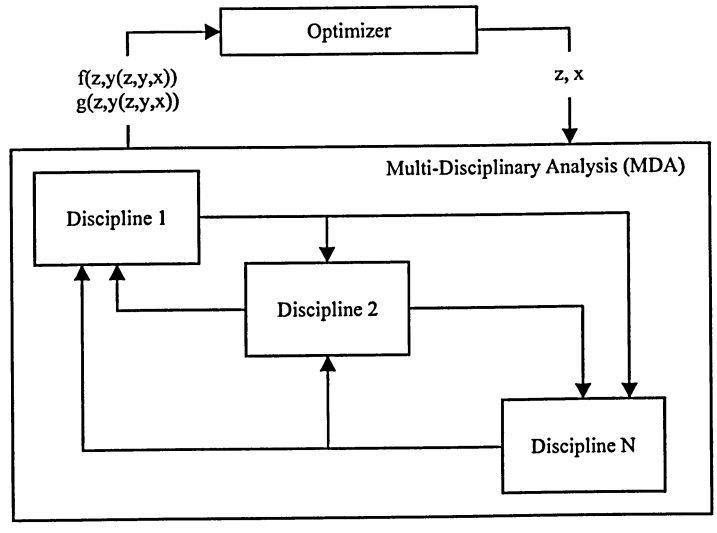


Figure 2-1: Multi-Disciplinary Feasible architecture [23]

optimizer. The optimizer in turn will output new global design variables and local design variables for use in the system analysis. The mathematical formulation of this architecture can be written as,

$$\begin{aligned} \min_{z,x} \quad & f(z, y_i(x, y_j, z), x) \qquad i = 1,\ldots,n \qquad j \neq i \\ s.t. \quad & g(z, y_i(x, y_j, z)) \leq 0 \end{aligned} \tag{2-1}$$

where f is the objective function and g are the constraint(s) of the problem. The terms z, y, and x are the global, coupling, and local design variable vectors respectively. The subscripts i and j represent indices that represent each discipline. It can be seen here that a method of solving non-linear systems is necessary because the coupling variables (y) are functions of the other coupling variables in the system. As mentioned previously, a popular method of solving non-linear systems is the Gauss-Seidel method.

A primary advantage of the MDF architecture is that on each iteration multidisciplinary feasibility is achieved by only treating global and local design variables as optimization variables [15], and leaving the coupling variables to be solved by the MDA. This forces that each iteration of the optimization will provide a feasible design. Another advantage of this method is that it is easy to implement provided that a solution of the non-linear coupled equations is possible. When this method is used for small to medium sized engineering problems the simplicity of the method can be taken advantage of [14]. The simplicity of the method is advantageous because implementation can be

achieved quickly and solutions obtained. However, there are several disadvantages of this method when used for large engineering problems. The main disadvantage of this method is that it is computationally expensive. This is because a MDA must be completed for every iteration of the optimization and for large problems this can add a significant amount of computational time. Depending of the solution method of the MDA an optimal feasible point may not always be found. An incorrect starting point for the MDA may lead to a non-optimal point and the optimizer may converge to a solution in the wrong direction. This problem is inherent in the Gauss-Seidel method, but can be avoided by implementing more sophisticated numerical methods. A numerical example of the MDF method can be found in Appendix A.

2.1.2 Individual Discipline Feasible

The Individual Discipline Feasible (IDF) method is an alternative to the possibly computationally expensive MDF method because it avoids a complete MDA upon each iteration. Instead of achieving multidisciplinary feasible design at every iteration of the optimization, IDF enforces individual discipline feasibility for each discipline and iteration. It only achieves a complete multidisciplinary feasible design when the solution has converged. The architecture of IDF shown in Figure 2-2 links a single optimizer and a decoupled disciplinary analysis for each discipline.

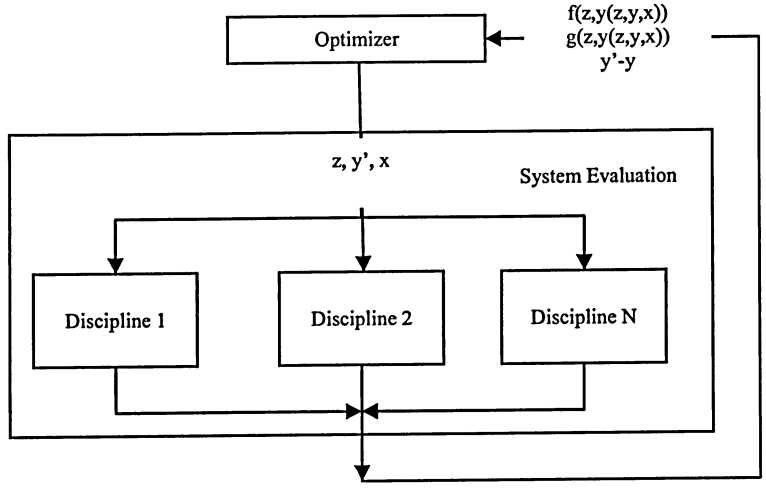


Figure 2-2: Individual Discipline Feasible architecture [23]

An important aspect of this method is that each discipline has been decoupled and now can run concurrently. This can help improve the computation performance of the optimization because there is no need for each discipline to communicate during the system evaluation. The change in the architecture forces the introduction of surrogate design variables that will allow the coupling variables (y) to be explicitly optimized by the optimizer [12]. Essentially, the coupling variables are promoted to be design variables for use within the optimizer. The inclusion of these new design variables changes the mathematical formulation to differ from the MDF as shown below,

$$\begin{aligned} \min_{z,x} \quad & f(z, y_i(x, y_j, z), x) \qquad i = 1,\ldots,n \qquad j \neq i \\ s.t. \quad & g(z, y_i(x, y_j, z)) \leq 0 \\ & y_i' - y(x, y_j', z) = 0 \end{aligned} \tag{2-2}$$

where f is the objective function, g are the constraint(s), and y'_i - y is the new auxiliary design constraint. The extra constraints must be included in order to drive the optimization to a feasible design upon convergence. Overall, the optimizer selects the surrogate design variables (y_i'), the global (z), and local (x) variables. These are then introduced to each discipline for analysis where the actual values of the coupling variables (y) can be calculated. The goal of this is to drive the surrogate design variables chosen by the optimizer to be equal to the coupling variables calculated by each discipline.

The advantage of this method is that a complete MDA is avoided upon each iteration which can drastically improve the necessary computational time. However, this is not always the case and there are times when this method may actually take longer than the MDF method. This is mainly because of the extra burden of satisfying the auxiliary design constraints. This method may also take longer when there are many coupling variables to consider. A numerical example of the IDF method can be found in Appendix A.

2.1.3 Collaborative Optimization

Collaborative Optimization (CO) is a decentralized bi-level optimization architecture that uses multiple optimizers to achieve an optimized multidisciplinary design. It is often used for loosely coupled large scale optimization problems because it

states the problem such that autonomy of each individual discipline is achieved [19]. As shown in Figure 2-3 the architecture consists of a single top level optimizer and an optimizer for each discipline. The top level (system level) optimizer is used to optimize the global (z) and coupling (y) variables while the discipline specific optimizers, or subspace optimizers, are used to optimize for global (z), coupling (y'), and local (x) design variables. The top level optimizer is also used to satisfy the interdisciplinary compatibility constraints which help achieve a feasible design upon convergence. The subspace optimizers are used to achieve interdisciplinary compatibility between the disciplines subject to local discipline constraints. The subspace optimizers are then connected to the discipline analysis so the design variables can be evaluated.

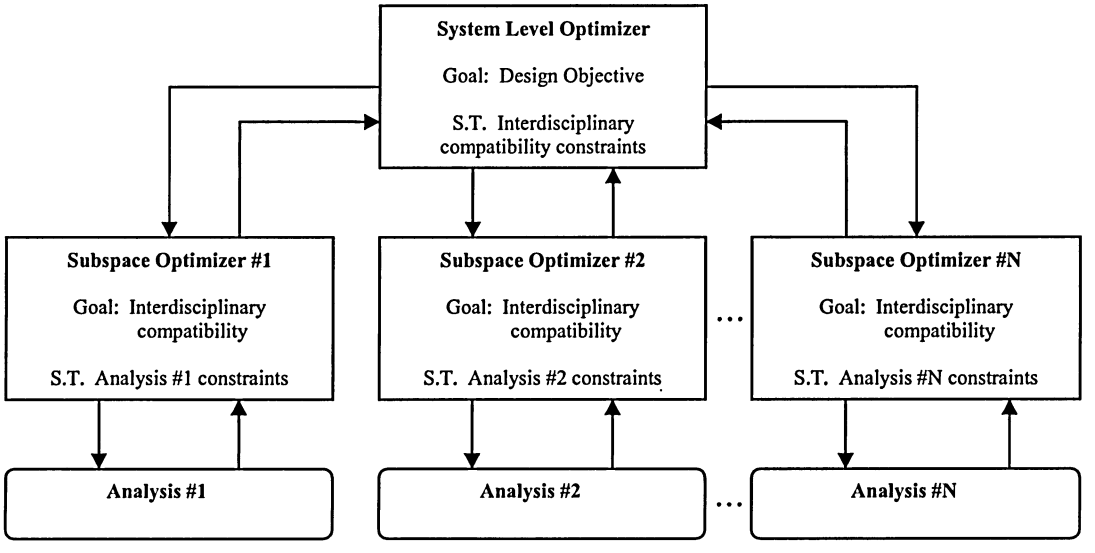


Figure 2-3: Collaborative Optimization architecture [23]

The mathematical formulation of the architecture for the system level optimizer is given as,

$$\begin{aligned} &\min_{z_{SL},x_{SL}} && f(z_{SL},y_{SL}) \\ &s.t. && J_i(z_{SL},z_i^*,y_{SL},y_i^*(x_i^*,y_j,z_i^*))=0 \qquad i=1,...,n \qquad j\neq i \end{aligned} \tag{2-3}$$

where f is the system level objective function and J_i is the interdisciplinary compatibility constraint for subspace i. The system level optimizer uses the overall design objective as its objective function. For example, this could be to minimize the overall system weight or maximize the overall aircraft range. The constraints are simply the values of the

interdisciplinary compatibility objective function values taken from the subsystems as seen in the subsystem formulation below,

$$\min_{z,y,x} \quad J_i(z_{SL}, z_i, y_{SL}, y_i(x_i, y_j, z_i)) = \sum (z_{SL} - z_i)^2 + \sum (y_{SL} - y_i)^2 \qquad \text{(2-4)}$$
$$s.t. \quad g_i(x_i, z_i, y_i(x_i, y_j, z_i)) \le 0$$

where J_i are the interdisciplinary compatibilities and g_i are the local constraints specific to each discipline at defined by i. At the subspace level the goal is it to minimize the interdisciplinary compatibility values.

The system level design variables z_{SL} and y_{SL} are chosen by the system level optimizer to minimize the overall objective function. These design variables are then passed to the subsystems to be used as target values for each subspace. Each subspace optimizer then optimizes for x_i, y_i, and z_i while trying to make the value of J_i as small as possible. This value of J_i is then passed back to the system level to be used as the equality constraint. As mentioned previously the goal is to achieve,

$$\sum J_i = 0 \qquad \text{(2-5)}$$

In other words, the system level selects values that will minimize the local objective function and each discipline will optimize each discipline completely independently. Using interdisciplinary constraints, each discipline will all find the exact same optimal design values upon convergence even though they do not directly communicate. For large complicated examples satisfying the equality constraint may take many iterations and it is quite possible that convergence will never be achieved. An alternative is to simply use inequality constraints [11] such as the one shown below,

$$\sum J_i \le 0.00001 \qquad \text{(2-6)}$$

This will allow that each discipline not necessarily find the exact same design point; they only need to be reasonably close.

CO offers many benefits over MDF and IDF because it offers completely autonomous discipline analysis. This means that each discipline analysis is independent of one another and do not need to pass data back and forth. This fits well into any industrial organization structures because each discipline analysis can be made by discipline experts and computations done on separate computers and passed to a master user to run the optimization routines [18]. A drawback of this method is that it takes

many more discipline analysis calculations and system level optimization iterations to achieve convergence. This can be offset, to some degree, by using separate computers for each discipline analysis so each discipline can be solved concurrently. A numerical example of CO can be found in Appendix A.

2.1.4 Concurrent Sub Space Optimization

The next method is Concurrent Sub Space Optimization (CSSO). This is similar to CO in that the disciplines are decoupled and run completely independently of each other. Another similarity is that there is system level optimizer and an optimizer for each local discipline. This is where the similarities end as can be seen in the system architecture diagram shown below in Figure 2-4.

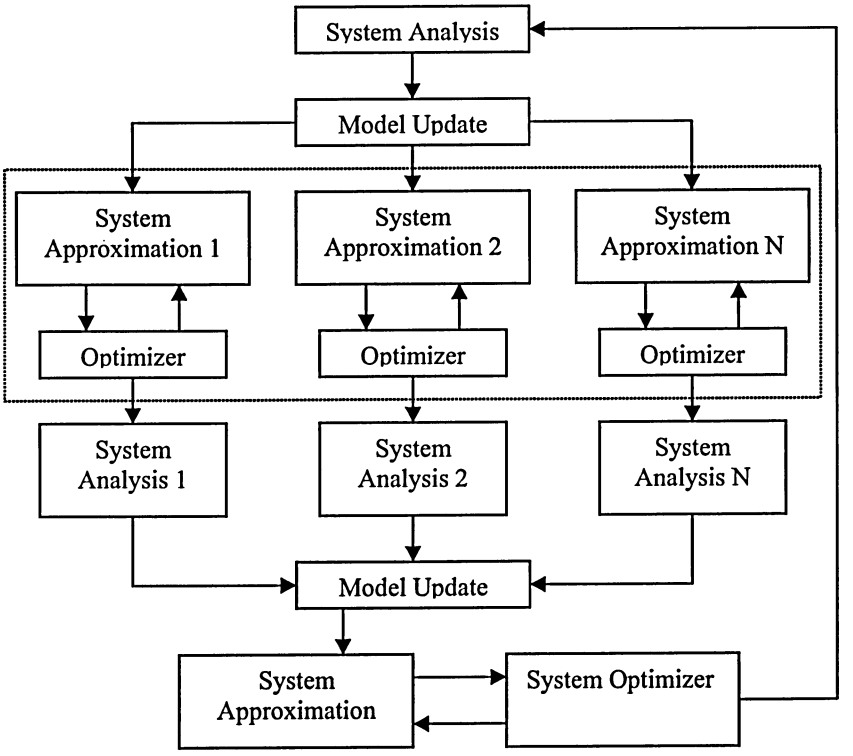


Figure 2-4: Concurrent Sub Space Optimization architecture [23]

For this method the initial guess is made and the optimization starts at the bottom of Figure 2-4 at the system optimizer. The initial optimization with the given starting points is completed and a system analysis is done to evaluate the effect of each the design variables and also to maintain feasibility. At this point a model is updated which will

provide information to the subspace optimizers about non-local design states. Essentially the model provides valuable information to the other optimizers about how changes in global design variables will affect other disciplines. These models are usually created using response surfaces that model the hyper-surface of design space. Once each subspace optimization is completed a system analysis is done for each discipline to analyze the effect the changes to the design variables had. This valuable information is then used to update the response surface model. Once the final model update is made the process starts all over again. An important aspect of this method is that the approximation models will always get better the longer the optimizer goes. This is because the response surfaces are always being improved. The mathematical model of CSSO can be seen next,

$$\begin{aligned} \min_{z,y^{app}} \quad & f(z,y^{app}) \\ s.t. \quad & g(z,y^{app}) \le 0 \end{aligned} \tag{2-7}$$

It can be seen here mathematically that the global objective function is calculated using the approximated coupling variables from the response surfaces. The same is done for the constraints. The sub space problem seen in equation (2-8) shows that the system analysis is completed using the approximated coupling variables for other disciplines.

$$\begin{aligned} \min_{z,y} \quad & f(z,y(x_i,y_j^{app},z_i),y_j^{app}) \qquad i=1,\ldots,n \qquad j \neq i \\ s.t. \quad & g(x_i,z,y_i(x_i,y_j^{app},z_i),y_j^{app}) \le 0 \end{aligned} \tag{2-8}$$

This can be seen when examining the sub space constraints. The local evaluation of the coupling variables (y_i) is a function of the approximated coupling variables from the other disciplines (y_j^{app}).

2.1.5 Bi-Level Integrated System Synthesis

Bi-Level Integrated System Synthesis (BLISS) is an MDO architecture that utilizes system decomposition the same as both CO and CSSO. It separates a large system that has many design variables into smaller subsystems that individually contain fewer design variables. The subsystems are optimized based on their own individual discipline specific variables, constraints, and objective function. The system and subsystem optimizations alternate and are linked via sensitivity analysis which allows a

design improvement upon each iteration. The type of sensitivity analysis is similar to the response surfaces that are used for CSSO except that they are calculated explicitly. The architecture of BLISS can be seen in Figure 2-5. The first step in the BLISS analysis is

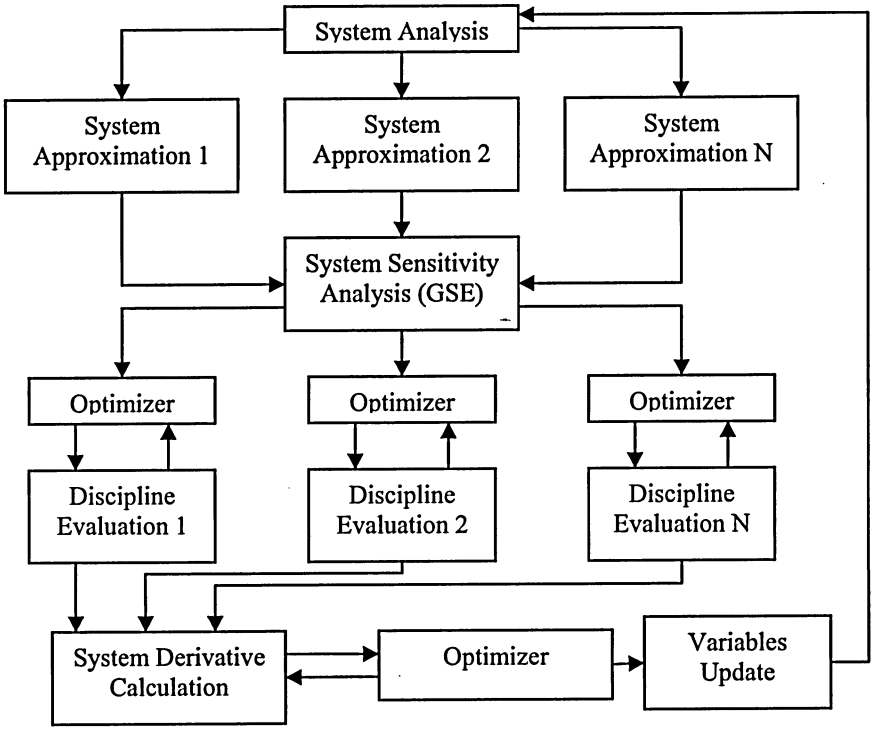


Figure 2-5: Bi-Level Integrated System Synthesis architecture [23]

the initialization of all of the global and local design variables. The initial values of the design variables need to be in the solutions feasible region in order to obtain a solution from the optimization that is feasible. The next step is to complete the full system analysis and the subsystem analysis for each discipline. The next step is to update the global sensitivity equations (GSE). The global sensitivity equations are a method of keeping track of how changes in design variables will affect other variables. The GSE's were first introduced by Sobieski [24] in order to perform accurate sensitivity analysis. This type of sensitivity analysis is required so when the subsystems are independently optimizing they can obtain critical information about how the local design variables are affecting other coupling variables in the other disciplines. The GSE's are similar to the response surfaces used in CSSO except that the GSE are exact calculations of the coupling interactions and response surfaces are only approximations. The method of calculating these sensitivities is critical for the efficiency of the optimization [23]. Once

each discipline has finished its own optimization a system derivative calculation can be performed for use in the system level optimizer. The final step is to update all of the design variables for use in the next loop. One of the most important parts of BLISS is that there can be human intervention upon each BLISS loop completion. This allows the user to review that the data from the previous iteration to decide if another BLISS loop should be done. This can also allow the user to change some values that may be causing problems either for derivative calculation or simply just discipline evaluation.

The mathematical formulation of this method is different than all of the other MDO methods that have been discussed so far. It can be seen from equation (2-9) that the traditional optimization problem has now been replaced by a derivative based formulation. This type of setup requires extensive knowledge of the method and lots of experience in the field of MDO.

$$\begin{aligned} \min \quad & d(f,x_i)^T \Delta x_i \\ s.t \quad & g_i(x_i) \le 0 \end{aligned} \tag{2-9}$$

The term $d(f,x_i)^T \Delta x_i$ represents the first order predicted objective function due to the change in x_i. The subsystem formulation is also derivative based as well.

$$\begin{aligned} \min \quad & \Phi = d(y_{1,i},x_1)^T \Delta x_1 + d(y_{1,i},x_2)^T \Delta x_2 + d(y_{1,i},x_3)^T \Delta x_3 + \ldots \\ s.t \quad & g(z,y(x,z),x) \end{aligned} \tag{2-10}$$

The objective function for BLISS is to make sure the change in the derivative term in equation (2-10) does not change from one iterate to the next. Overall, the BLISS method is still a relatively new method for doing MDO and many of the aspects of the research are still being developed. At this point in its development it has been seen that BLISS is a method that should be used for detailed large scale industrial optimization problems [25].

2.2 MDO in Aircraft Design

There has been a considerable amount of research performed in the aerospace field regarding MDO. This is simply because of the inherent complex nature of the aerospace industry. As shown in Figure 2-6 an aircraft consists of many different

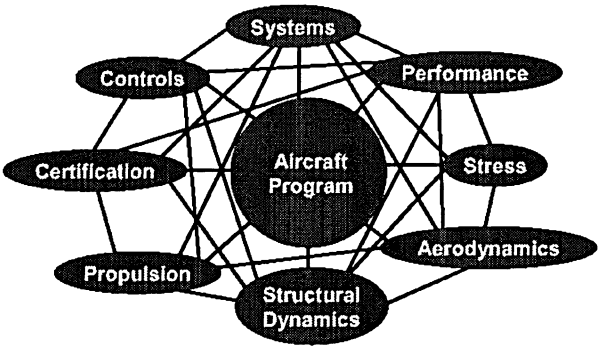


Figure 2-6: The many different disciplines in aircraft design [23]

coupled disciplines. Taking into consideration all of these interactions at any stage of the design process can be time consuming. It is difficult to find the optimal design point when all other disciplines directly affect the other disciplines being considered. For this reason MDO originated in the aerospace field and has started to gain popularity in recent years. Extensive survey papers by Sobieski and Haktfa [8] and Holinger et al. [26] do a good job of reviewing the current needs and trends that MDO in the aerospace industry is taking. Sobieski and Haktfa concluded that the two main obstacles for MDO are the high computational demands and organizational challenge of MDO itself. A key aspect of their paper was that they identified that there is very little communication between aerospace multidisciplinary optimization research and other engineering research communities. For example, MDO is capable of being used in both chemical and electrical engineering applications but its benefits are taking longer to be recognized. Holinger et al. [26] identified the major technology gaps in the MDO industry. It was reported from interviews that most aerospace industry people would like MDO methods that are easy to implement as well as methods in which the optimization can visualized in CAD software.

There has been extensive research in this field using low-fidelity methods for preliminary aircraft design. Kroo et al. [18] displayed how a relatively simple model for preliminary aircraft design can be used with the collaborative optimization method to find an overall optimal design point when considering three disciplines. Using a low-fidelity aircraft model Perez et al. [23] successfully compare the five major MDO methods

against one another. Perez et al. concluded that amount of coupling and size of the optimization problem directly influences the MDO methodology that should be used.

2.3 Optimization Methods

The optimization procedure that is used within this work is sequential quadratic programming (SQP). SQP was chosen because it is currently the most popular gradient based optimizer used for this type of optimization. In order to fully explain the important elements of SQP it is important to begin with a much simpler optimizer. A simpler version of SQP is sequential linear programming (SLP). It works the same as SQP except that linear approximations of the functions are used.

2.3.1 Sequential Linear Programming

Linear programming (LP) techniques and their solution methods have been extensively studied and have become extremely effective methods of solving optimization problems. They have become popular because they can optimize a large number of design variables with very low computation expense. Linear programming can be used even to solve non-linear optimization problems subject to non-linear constraints. In order to adapt linear programming to solve non-linear problems a method known as sequential linear programming (SLP) is used. SLP is a method of approximating the solution of a non-linear optimization problem using quickly solved linear sub-problems. The approximations can be extremely accurate and a well programmed SLP can sometimes outperform more complicated optimization algorithms [27]. For higher order functions the approximation may become inaccurate and can only be made accurate when the functions are evaluated near the optimal design point. To help decrease the inaccuracies due to evaluation points away from the optimal, an adaptive limiting strategy proposed by Lund et al. [28] can be used, that forces the maximum change of 5% away from the current design point with each iteration. This results in a more accurate solution and one that is less likely to diverge.

SLP is generally used when the objective function is non-linear, and one or more of the constraints (equality or inequality) have a non-linear relationship [29]. If the

constraints are completely linear the Jacobian cannot be updated upon each iteration and the step direction cannot be determined.

2.3.1.1 Algorithm Development

The algorithm of SLP is a simple extension of Taylor's series that ignores higher order terms and keeps the linear ones. From a simple problem formulation such as:

$$\begin{aligned} &\min && f(x), \\ &s.t. && h_k(x) = 0 \qquad k = 1,2,....,l \\ &&& g_j(x) \le 0 \qquad j = 1,2,....,m \end{aligned} \tag{2-11}$$

where $f(x)$ is a non-linear objective function and $h_k(x)$ and $g_j(x)$ are non-linear constraints. The functions can be approximated using first order Taylor's series to linearize the objective function and constraints [29]. This expansion can be seen below:

$$\begin{aligned} &Minimize && \tilde{f}(\Delta x) : f(x_i) + \nabla f(x_i)^T \Delta x \\ &Subject\ to: && \tilde{h}_k(\Delta x) : h_k(x_i) + \nabla h_k^T(x_i)\Delta x = 0; \qquad k = 1,2,...,l \\ &&& \tilde{g}_j(\Delta x) : g_j(x_i) + \nabla g_j(x_i)^T \Delta x \le 0; \qquad j = 1,2,....,m \\ &&& \Delta x_i^{low} \le \Delta x_i \le \Delta x_i^{up}; \qquad i = 1,2,....,n \end{aligned} \tag{2-12}$$

where $\tilde{f}(\Delta x)$ is the expanded objective function, $\tilde{h}(\Delta x)$ is the expanded equality constraint, $\tilde{g}(\Delta x)$ is the expanded inequality constraint, Δx is the step size and direction of the next iteration and ∇ is the gradient function. The upper and lower bounds for the design vector Δx_i are Δx_i^{up} and Δx_i^{low} respectively. Calculation of Δx is found by solving a linear programming problem. The solution is typically done using the simplex method. The next value of x to be used on the following iteration is found simply by:

$$x_{i+1} = x_i + \Delta x \tag{2-13}$$

where x_i is the current iteration design point and x_{i+1} is the next iterations design point. At the end of each iteration the current design point is compared to the previous until a satisfactory error, ε, is computed.

The linear approximations that SLP makes using Taylor's Series can be seen visually in Figure 2-7. For this example there are two inequality constraints represented by g_1 and g_2. The objective function is to find the minimum value of x1 by changing the value of x2. The constraints can be linearly approximated by Taylor's series and

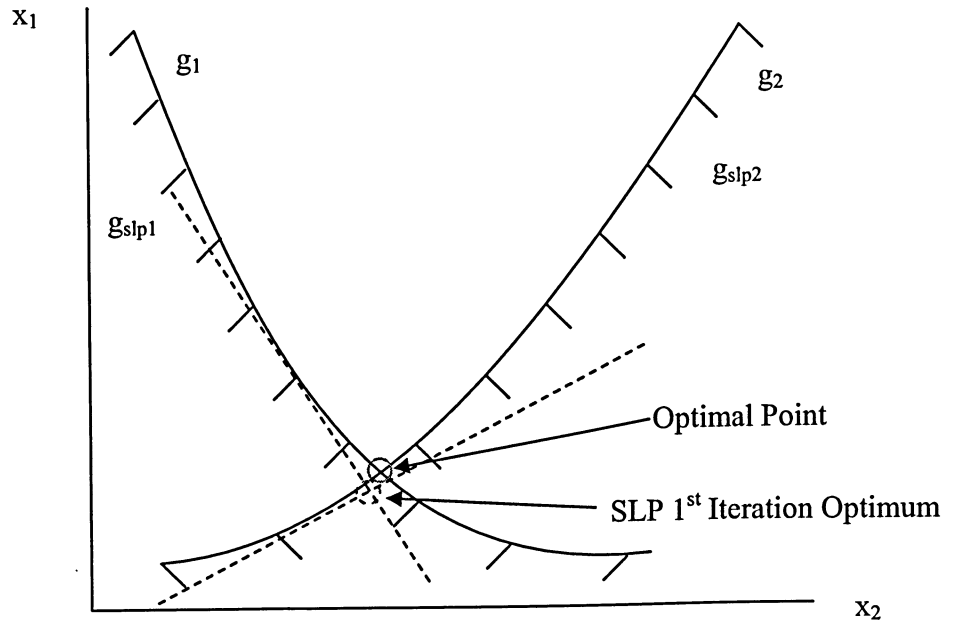


Figure 2-7: Visualization of SLP

can be represented by g_{slp1} and g_{slp2}. The intersection of these linear lines is the 1st iteration optimum point. Solving the linear programming problem will give a new point which is close to the optimal point. After a number of iterations the two approximating lines will eventually intersect near the optimal point and thus give an approximate solution. Note that for extremely non-linear functions with many hills and valleys the approximation may not work and a better initial guess will be required to find the global optimum.

SLP overall provides a good approximating technique for non-linear optimization problems because it will often converge rapidly [30]. Figure 2-8 shows a flow chart that details the several steps that are involved in solving SLP problems. The task of scripting this is rather simple if an external linear programming solver is available, otherwise the implementation of this is not trivial because a general linear programming algorithm must be scripted.

2.3.2 Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is an iterative version of quadratic programming (QP) that is popular and widely used. It can be used to solve non-linear

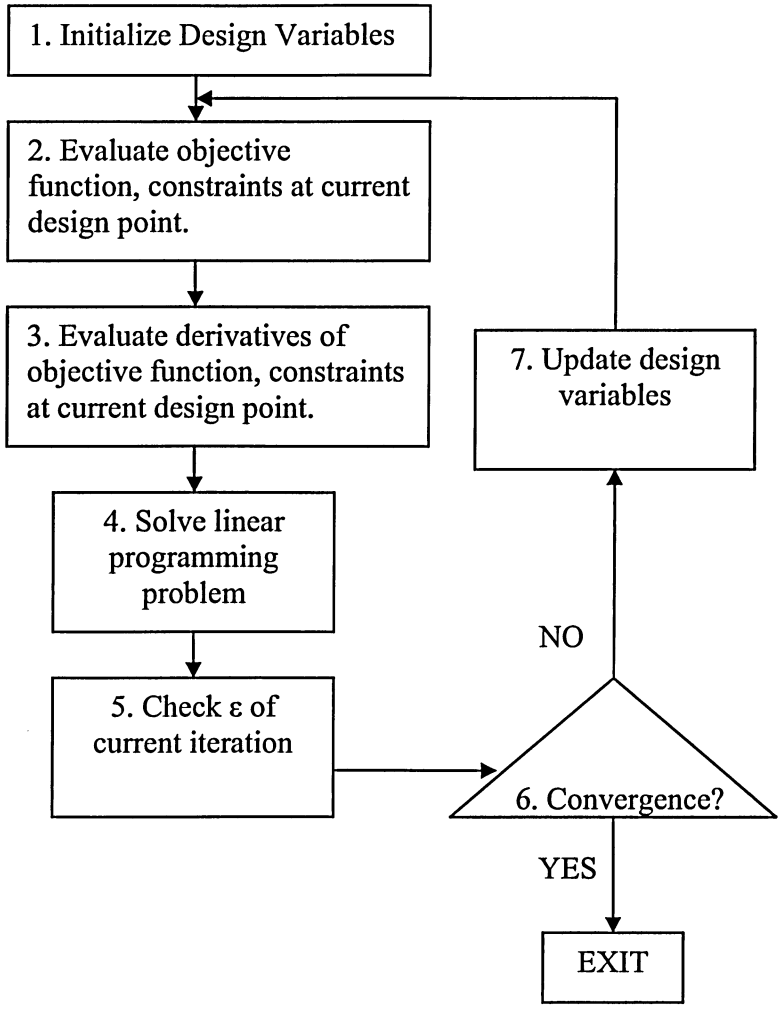


Figure 2-8: Flow chart representation of SLP

programming problems that have either linear constraints or non-linear constraints. This method of solving non-linear optimization problems is more accurate than SLP when higher order problems are considered. This is because a better approximation of the non-linear function can be obtained. Implementation of SQP into code is not trivial and will be discussed further here. An example [31] of a problem that can be solved using SQP is,

$$\begin{aligned} &\textit{Minimize} && f(x_1,x_2): \; x_1^4 - 2x_1^2x_2 + x_1^2 + x_1x_2^2 - 2x_1 + 4 \\ &\textit{Subject to}: && h(x_1,x_2): \; x_1^2 + x_2^2 - 2 = 0 \\ & && g(x_1,x_2): \; 0.25x_1^2 + 0.75x_2^2 - 1 \le 0 \\ & && 0 \le x_1 \le 5; 0 \le x_2 \le 5 \end{aligned} \tag{2-14}$$

where the term $h(x_i)$ represents equality constraints and $g(x_i)$ represents inequality constraints and $f(x_1,x_2)$ is the objective function. Each constraint term can be expanded to include l number of equality constraints and m number of inequality constraints. The number of design variables x_i ranges from $i=1,2,...,n$. This example clearly has a non-linear objective function as well as non-linear equality and inequality constraints. Solution of a general system using SQP will be discussed further in the following section.

2.3.2.1 Algorithm Development

Describing the process of how SQP works requires many definitions that may be trivial for some but it repeated here for completeness. The general optimization problem is:

$$\begin{aligned} \min \quad & f(x), \\ s.t. \quad & h_k(x) = 0 \qquad k = 1,2,....,l \\ & g_j(x) \le 0 \qquad j = 1,2,....,m \end{aligned} \tag{2-15}$$

where $f(x)$ is a non-linear objective function and $h_k(x)$ and $g_j(x)$ are non-linear constraints. The functions can be expanded using second order Taylor's series on the objective function and approximate the constraints by using first order Taylor's series to linearize the constraints. This expansion can be seen next:

$$\begin{aligned} \textit{Minimize} \quad & \tilde{f}(\Delta x) : f(x_i) + \nabla f(x_i)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x_i)\Delta x \\ \textit{Subject to:} \quad & \tilde{h}_k(\Delta x) : h_k(x_i) + \nabla h_k^T(x_i)\Delta x = 0; \qquad k = 1,2,...,l \\ & \tilde{g}_j(\Delta x) : g_j(x_i) + \nabla g_j(x_i)^T \Delta x \le 0; \qquad j = 1,2,....,m \\ & \Delta x_i^{low} \le \Delta x_i \le \Delta x_i^{up}; \qquad i = 1,2,....,n \end{aligned} \tag{2-16}$$

where $\tilde{f}(\Delta x)$ is the expanded objective function, $\tilde{h}(\Delta x)$ is the expanded equality constraint, $\tilde{g}(\Delta x)$ is the expanded inequality constraint, Δx is the step size and direction of the next iteration and ∇ is the gradient function. The upper and lower bounds for the design vector Δx_i are Δx_i^{up} and Δx_i^{low} respectively. Calculation of Δx is found by solving a QP problem. The next value of x to be used on the following iteration is found simply by:

$$x_{i+1} = x_i + \Delta x \tag{2-17}$$

where x_i is the current iteration design point and x_{i+1} is the next iterations design point. At the end of each iteration the current design point is compared to the previous until a satisfactory error, ε, is computed.

The method of solving a QP problem has been researched extensively. Both Zhu [32] and Shanno et al. [33] use rather traditional methods in that they solve the Karush-Kuhn-Tucker (KKT) QP problem. The generalized QP sub-problem problem as described by Zhu is:

$$\begin{aligned} \min \quad & \nabla f(x)d + \frac{1}{2}d^T H d \\ .s.t. \quad & g_j(x) + \nabla g_j(x)^T d = 0, \qquad j = 1,2,....,m. \end{aligned} \tag{2-18}$$

Where d is the search direction for the next iteration; which is equivalent to Δx, ∇f is a vector is first derivatives of the objective function, and ∇g_i is a vector of the first derivatives of the constraints. Note that here the inequality constraints have been changed into equality constraints by using slack variables. H is the Hessian, which is a matrix of the partial second derivatives of the objective function with respect to all of the design variables, and it must be symmetric and positive definite. Positive definite means that the eigenvalues of the Hessian must be greater than zero. The Hessian is calculated in general using finite differencing and is defined by Nocedal et al. [34] as:

$$H = \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{2-19}$$

where f must be at least twice continuously differentiable. At each new iteration point a new Hessian must be calculated. The Hessian itself can be updated by calculating the entire matrix again, which does not ensure that it is positive definite at every single design point and thus it is not recommended. To make sure that the matrix is always positive definite it can be updated using the Broydon-Fletcher-Goldfarb-Shanno (BFGS) method. The BFGS method uses the first Hessian to create a current Hessian. It ensures that future Hessians will remain positive definite as long as the first Hessian is positive

definite [31]. In order to solve this QP the KKT matrix can be made that will allow the problem to be solved by a system of linear equations. The KKT matrix is defined by Nocedal et al. as:

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ c \end{bmatrix} \tag{2-20}$$

where

$$c = Ax - b, \qquad g = d + Hx, \qquad p = x^* - x \tag{2-21}$$

where A is a Jacobian of the constraints, λ^* is a vector of the Lagrange multipliers, x is the current estimate of the solution, b is the constraint limits, p is the current step vector, and x^* is the next iterations estimated solution. The solution of the KKT matrix will solve for the solution of the current condition. See Figure 2-9 for flow chart of SQP solution process.

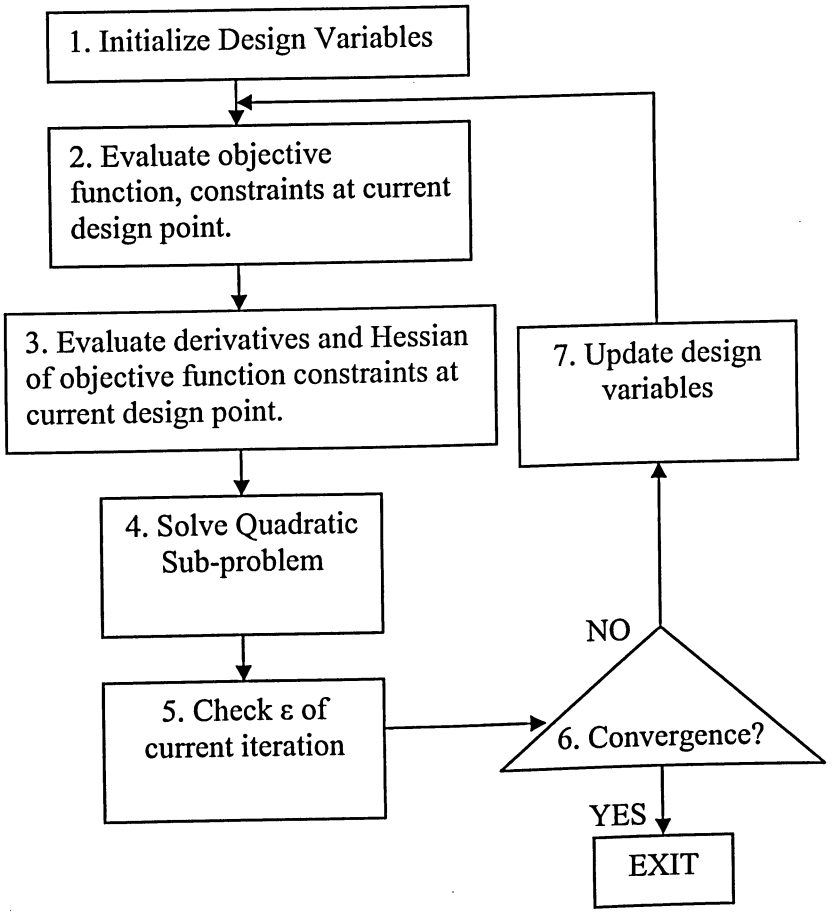


Figure 2-9: Flow chart representation of SQP

SQP is powerful because it can handle both equality and inequality constraints. This is due to the Lagrange multiplier that allows the constraints to be added linearly to the objective function. A SQP solver is commercially available in many mathematical software packages. The SQP solver that is used for this research is available as a toolbox for MATLAB. Several researchers have successfully used it to perform complex optimizations [35, 36].

Chapter 3

High-Fidelity Aircraft Optimization

3.1 Current Research

Multidisciplinary design optimization is steadily becoming a design standard within the aerospace industry because of the need to remain competitive by reducing costs and improving system performance. In order to improve the designs and reduce the redesign costs of components different levels of fidelity have been incorporated into MDO. Using higher fidelity models can allow components to be designed in high detail with very little human interference. Interest in increasing the level of design fidelity has been growing in parallel with the advent of MDO methodologies. The level of fidelity, specific to the aerospace industry, can be broken down into three main levels [37]. Level 1 fidelity or low-fidelity uses empirical models; level 2 fidelity, or intermediate-fidelity, uses primarily beam theory and aerodynamic panel methods to estimate the system reactions and sensitivities. Level 3 fidelity, or high-fidelity, uses principally computational fluid dynamics (CFD) and the finite element method (FEM) during system analysis.

Low/Intermediate fidelity analysis has been performed by Martins [38]. He used a simple beam structure for the wing box and a simple panel method for the aerodynamics to show the coupling between structures and aerodynamics. Recently, a new approach to conceptual aircraft design has been introduced by Perez et al. [39, 40] where low-fidelity aerodynamic and aircraft dynamics models have been used in

combination with control theory. Perez et al. were able to show that introducing control theory and aircraft dynamics to the optimization the performance of the aircraft mission could be enhanced.

Intermediate-fidelity analysis combined with MDO is commonly implemented for aircraft design because it can give an approximation of the optimal aircraft configuration rather quickly. Sobieski et al. [20] performed an intermediate-fidelity analysis to show the effectiveness of the BLISS MDO architecture. The same model has been used by others to display the effectiveness of MDO architectures [23]. The Sobieski et al. [20] model uses three different disciplines that primarily focuses on the aerodynamic shape and sizing of a super sonic business jet. It contains a relatively simple analysis that used mostly analytic equations. Even though the analysis is mostly analytic, it gains recognition as an intermediate-fidelity method primarily because of the high number of design variables and high number of coupling variables that link all disciplines. Wakayama and Kroo [41] developed a process for multidisciplinary design optimization of blended-wing-body aircraft. This method, known as Wing Multidisciplinary Optimization Design (WingMOD), was an intermediate-fidelity problem. The aerodynamic loading was calculated using a vortex-lattice code and the structures were solved using a monocoque beam model. The code was capable of high-fidelity methods but only intermediate-fidelity was used because of the extremely long solving times for the high-fidelity CFD solution. Wakayama and Kroo indicated that CFD will be implemented later when the solving times decrease. In addition it was also mentioned that FEM was not an important aspect to their optimization because their intermediate structural analysis was just as good as FEM for general aircraft sizing. This point can be argued because in the industry experts tend to insist on using high-fidelity state-of-the-art modeling tools because of the level of detail that can be obtained [37].

High-fidelity multidisciplinary design optimization has been gaining popularity as a research topic for the past several years. This is primarily due to the increases in computational solving times of the high-fidelity design tools. The two most popular tools for high-fidelity design optimization in aerospace design are CFD and FEM. Many researchers have used both CFD and FEM in the optimization process. Gumbert et al. [42] has published several papers concerning the optimization of rigid and aeroelastic

wings. Originally Gumbert et al. developed a formulation of simultaneous aerodynamic analysis and design optimization (SAADO) for a rigid 3-D wing. In order to show the aerodynamic optimization process graphically using contour plots of the objective function only wing tip chord length and leading edge setback were used as design variables. In order to show the robustness of the formulation several optimizers were used and ultimately each found similar optimal points. The number of aerodynamic design variables was later increased from two to be fifteen [35]. The results showed that the aerodynamic optimization of a 3-D wing was possible using CFD and FEM. With the success of this research Gumbert et al. published several papers on flexible aeroelastic wings [35,43,44]. Their research was aimed at including both aerodynamics and structures into the design process. The formulation used the MDF-MDO architecture with SQP as the optimizer. In order to decrease the solving times of the multidisciplinary analysis the coupled steady-state-aero-structural field equations were not completely converged at every iteration. Only enough CFD iterations were done to allow the optimizer to obtain a good approximation of the gradient. In the end Gumbert et al. were able to decrease the computational time of the optimization by 50% compared to more traditional methods. Reuthers, Alonso, and Martins [38,45,46,47] were the authors of many papers concerning high-fidelity multidisciplinary optimization of complete aircraft configurations. They have introduced new couple-adjoint sensitivity methods that can decrease solving times. The coupled-adjoint method is possible because they can obtain the sensitivities directly from in house CFD codes. There are many more researchers that have conducted research in this area, too many to mention here [48, 49].

Although computational times for solving the expensive CFD codes is decreasing most researchers still use multiple levels of fidelity, or variable-fidelity, in their research to help speed up the solving times of the optimizations. The long CFD solving times are typically due to solving full Navier-Stokes equations with very fine CFD quality meshes. To decrease these solving times Euler equations are often used when solving the equations in the flow domain. This is popular because the Euler equations provide a good approximation of the flow even though the viscous properties of the flow are ignored [50]. The Euler equations contain fewer equations to solve and can provide a converged CFD solution faster than employing Navier-Stokes equations. Very little

MDO research has been completed where the use of the full Navier-Stokes equations were used to obtain pressure distributions. Another method of saving time is to increase the coarseness of the CFD mesh [42]. This can allow a CFD solution to converge more quickly when compared to a finer mesh because there are fewer equations to solve. Sometimes, CFD is avoided completely and vortex lattice methods are used instead. For example, Bhatia et al. [51] used a vortex lattice method to calculate the aerodynamic forces and used MSC/Nastran for the FEM analysis. Even though combining two different fidelity methods for CFD and FEM is quite common; researchers are now developing variable-fidelity methods in which both high density/low density CFD grids and Navier-Stokes/Euler can be used together in one single optimization [52,53].

Combining high-fidelity and intermediate-fidelity methods in order to reduce the optimization time is becoming more common even though the concept has been around for a while [54]. Recently many researchers have started to develop optimization routines that can effectively use a combination of fidelities to improve solving times. Alexandrov et al. [55] started to develop trust region methods for variable fidelity optimization for nonlinear programming. They used trust regions that determined if the intermediate-fidelity method was doing a good job of approximating the gradients. If the intermediate-fidelity method was determined to be doing a good job of calculating the sensitivities it is used for the next iteration, otherwise the high-fidelity model is used. This research was continued and applied to variable fidelity wing design by Alexandrov et al. and most recently by Le Moigne and Qin [53]. Both Alexandrov et al. and Le Moigne and Qin showed that optimization of wings can be done using variable-fidelity methods. Le Moigne and Qin used both coarse and fine CFD grids in combination with Euler and Navier-Stokes flows. They showed that the high-fidelity model can be used to correct the low-fidelity model. This way the low-fidelity model can be used to calculate the required gradients and periodically be updated by the high-fidelity model.

Developing an optimization routine for simultaneous CFD and FEM is challenging and obtaining a solution can take quite a long time. Le Moigne and Qin required 265 hours to complete their optimization. In order to decrease the solving time of this research two separate methods were used. The first was to only use Euler equations when solving the CFD as many others have done. The second method was to

increase the coarseness of the CFD mesh. This way the CFD solver does not have as many equations to solve and CFD convergence is obtained rather quickly.

Overall, many different levels of fidelity can be used to perform optimization and they can even be combined to provide faster solving times. The most important thing to realize is that regardless of the fidelity method used to obtain the responses, accurate calculations of the system sensitivities is required. If the derivatives are not accurate search directions (gradients) will not find the optimal solution.

3.1.1 Aeroelastic Coupling

In all aspects of high-fidelity aero-structural optimization the need for coupling the aerodynamics and structures is necessary. Sometimes it is called CFD/FEM coupling, or CFD/CSM coupling, or even fluid-structure interaction (FSI). Regardless of what the coupling is called it is essentially just computational aeroelasticity. Most of the available literature mentioned in the previous sections agree on how the coupling process should be done for this type of research. A good generalized approached was published by Samareh and Bhatia [56] that provides a unified approach to any multidisciplinary interaction between a solid and fluid. Their approach used the idea of a transformation matrix to change the aerodynamic pressure to structural loads. The interaction between the two disciplines can be modeled by,

$$\{F_2\} = [T_{21}]\{F_1\} \tag{3-1}$$

The $\{F_1\}$ vector could represent the aerodynamic pressures and $\{F_2\}$ vector could represent structural loads on the FEM model. The matrix $[T_{21}]$ is the transformation matrix between the two disciplines. The entire aeroelastic analysis is an iterative process that was expressed mathematically by Samareh and Bhatia as,

$$\{F_F\} = \{Flow\ \ Solution\ \ (G_F + \delta_F)\} \tag{3-2}$$

$$\{F_S\} = [T_{SF}]\{F_F\} \tag{3-3}$$

$$[K]\{\delta_S\} = \{F_S\} \tag{3-4}$$

$$\{\delta_F\} = [T_{FS}]\{\delta_S\} \tag{3-5}$$

The iterative process starts with equation (3-2) where $\{F_F\}$ are the aerodynamic forces and $\{G_F\}$ and $\{\delta_F\}$ are the CFD grid and deformations from FEM respectively. The

aerodynamic loads are calculated and then are used in equation (3-3). In equation (3-3) the matrix $[T_{SF}]$ is the transformation matrix between the structural and fluid disciplines. The forces that are transferred to the FEM model are held in the vector $\{F_S\}$. These forces are then used in equation (3-4) where the structural stiffness matrix, $[K]$, is used to calculate the structural displacements, $\{\delta_S\}$. Finally, the structural displacements are transferred to displacements on the CFD grid using equation (3-5). This process is done until the change from the previous iteration to the next iteration is less than the tolerance, ε. This can be seen mathematically as,

$$\{F_F\} - \{F_F\}^- \leq \varepsilon \tag{3-6}$$

$$\{\delta_S\} - \{\delta_S\}^- \leq \varepsilon \tag{3-7}$$

It can be seen from equation (3-6) and equation (3-7) that the convergence of the aeroelastic solution is obtained when the aerodynamics forces and the structural displacements do not change from one iteration to the next. Note here that the solution method here does not include the time domain which means that the converged solution is a static-aeroelastic solution. Using time along with MDO would force the optimization to take an extraordinary large amount of time. This process can be seen graphically in Figure 3-1 below.

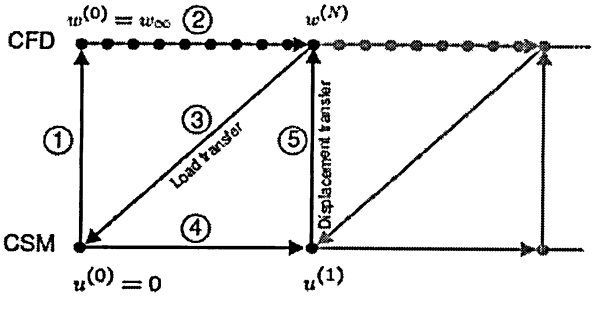


Figure 3-1: The CFD/CSM coupling procedure [47]

Graphically, the process starts by taking the CSM displacements and applying them to the CFD grid. The CFD solution is then solved and the loads are transferred to the CSM model. The displacements on the CSM are calculated and then used to change the shape of the CFD grid. This process continues until the static-aeroelastic convergence is found.

Sometimes the static-aeroelastic solution can be sped up so convergence is reached more quickly. Gumbert et al. [35], for example, showed that convergence of the

static aeroelastic problem can be done without having the CFD solution come to complete convergence at every CFD/FEM coupling iteration. It was found that it is only necessary for the CFD solution to converge completely for the final CFD/FEM coupling iteration. All previous CFD solutions need only an approximation of what the loads on the wing will be. Another method of decreasing the time required for static-aeroelastic convergence is using under relaxation methods as was done by Alonso et al. [47]. The method was originally used to remove the oscillations that occur during the convergence of the CFD/FEM coupling. It was found to result in absolute convergence sooner compared to the conventional method that uses the latest structural displacements to update the CFD grid. For the method used by Alonso et al. the displacement that is applied to the CFD grid is a combination of the current structural displacement and the displacement from the previous iterate. This can be seen mathematically as:

$$\delta_S^+ = \delta_S^- + \beta(\delta_S - \delta_S^-) \tag{3-8}$$

where β is the under relaxation factor that is used to scale the displacements. The terms δ_S^+, δ_S^-, and δ_S are the displacements of future iterate, previous iterate, and the current iterate. It was found that using this method, the static-aeroelastic convergence was obtained in about 10 iterations with no oscillations [47].

3.1.2 CFD/FEM Mesh Mapping

Coupling between two different software packages or analysis codes is not as trivial as what is described in the previous section. In order to successfully transfer loads or displacements from one grid to the next it is necessary to map one mesh onto the other. This can be difficult because the discretization of the CFD and FEM models are never the same. It must be determined which node from one mesh should be mapped to another node of the other mesh. There are several different methods to map one mesh onto another. One of the most common methods of coupling the CFD and FEM meshes is to use finite interpolation elements. Both Ahrem et al. [57] and Beckert [58] have used interpolation elements in order accurately determine the transfer of loads and displacements. Figure 3-2 shows how the finite interpolation element works. Starting from the CFD grid point on the top of the diagram the pressure information is projected

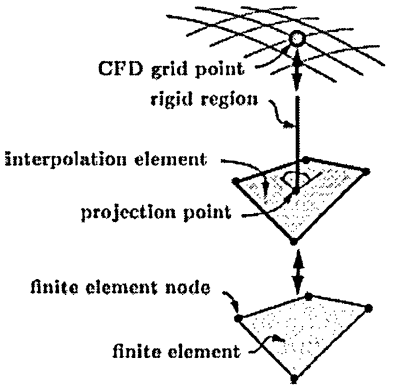


Figure 3-2: The finite interpolation element [58]

to the interpolation element. Using bilinear shape functions the data is then transferred to the finite element nodes. The process also works in the other direction where the finite element nodal displacements are transferred to the CFD grid.

There are several other methods that can be used to transfer information between meshes. Bhardwaj [59] summarized several different methods that can be used for transformations; these are finite-plate spline, multiquadratic-biharmonic scheme, thin-plate spline, inverse isoparametric mapping, non-uniform B-splines, and inifite-plate spline. Each of the aforementioned interpolation methods each have there pros and cons and one method is not necessarily better than the rest. The method that has the most promise for this type of application is non-uniform rational B-splines (NURBS). NURBS is essentially a method of obtaining a continuous surface of any property in three or four dimensions. This could be either pressure distribution or displacement distribution. The benefits of this type of method are that it is continuous and therefore allows data to the transferred directly to any node on a model.

Chapter 4

Aero-Structural Analysis

4.1 Model Geometry

There are two distinct geometry models that need to be created for the aero-structural analysis. The first model is used to determine the aerodynamic loads acting on the wing. The second is used for calculating the structural response of the wing due to the aerodynamic loading. Because these models are used during the optimization process they must be able to automatically regenerate themselves for any possible wing configuration. In addition, they must also be compatible with each other in order accurately determine the aeroelastic effects.

4.1.1 NACA 4-Digit Airfoil

The airfoil shape used for this optimization was chosen to be the NACA 4-digit airfoil because of the simplicity of the geometry. A more complicated airfoil shape would have caused automatic model generation to be more difficult. It was also decided that in order to keep the number of design variables limited, the airfoil, was to remain constant along the span of the wing. Typically, commercial aircraft use many different airfoil shapes along the span of the wing. For future optimizations, it would be interesting to keep the wing geometry constant while optimizing several different airfoil shapes along the wing span.

The NACA 4-digit airfoil requires three different pieces of information to define the airfoil shape. The three different parameters can be seen here,

NACA 4-Digit Series:

4	4	12
Max camber in % chord	Position of max camber in 1/10 of chord	Max thickness in % of chord

where the nomenclature that is used for these definitions can be found in Figure 4-1 taken from Ref. [50].

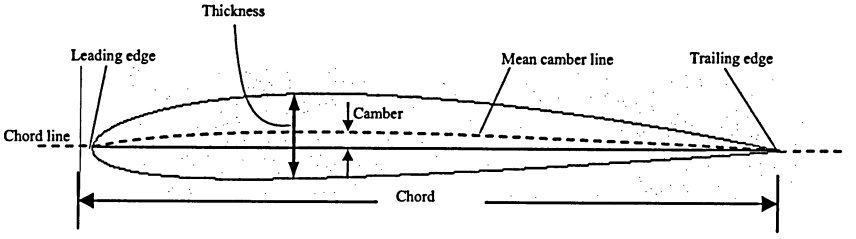


Figure 4-1: Typical airfoil nomenclature

For general airfoil definitions the mean camber line is defined as the locus of points that sits halfway between the upper and lower surface of the airfoil. The camber is defined as the maximum distance from the chord line to the mean camber line. The thickness of the airfoil is defined as the maximum distance from the upper surface to the lower surface of the airfoil.

This work required that the airfoil must be generated automatically as required by the optimizer and the airfoil data points to be exported for use in multiple CAD packages. To do this, a function was created that can be used to create any NACA 4-digit airfoil. The first step in this process is to generate a symmetric airfoil that does not have any camber using the following equation [60]:

$$y = \pm(t/0.2)\cdot\left(0.2969x^{0.5} - 0.126x - 0.3537x^2 + 0.2843x^3 - 0.1015x^4\right) \quad \textbf{(4-1)}$$

Given a set of data points defined by x, all of the data points for the airfoil thickness can be generated. Note that in order to accurately model the airfoil shape, a bias must be used to increase the number of data points near the leading edge. The next step is to generate a parabola that extends from the leading edge to the position of maximum camber. A second parabola must also be generated that extends from the position of

maximum camber to the trailing edge. The parabolas are defined by two variables, x_c and y_c which can be seen in Figure 4-2.

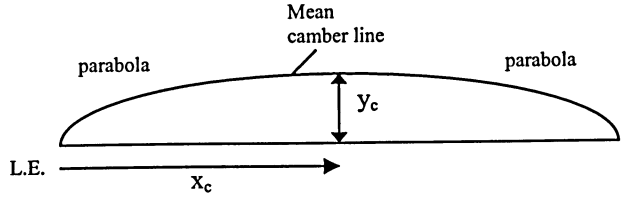


Figure 4-2: Definition of NACA 4-digit mean camber line

The variable x_c is the position of the maximum camber and y_c is the value of the maximum camber. Once the data points are generated for the mean camber line, they are simply added to the data points evaluated from equation (4-1). The result is an airfoil that can be easily regenerated as needed and is robust enough to include different chord lengths and thicknesses.

4.1.2 Parametric Aerodynamic Wing Shape

The wing shape that is used for the optimization has been parameterized to give the optimizer as much freedom in choosing the aerodynamic shape as possible. This was done by selecting several design variables that would have the greatest effect on the aerodynamics of the wing. The different parameters that are used during the optimization are defined in Figure 4-3. The first parameter that is used to define the wing shape is the leading edge wing sweep, Λ. This was chosen as a design variable because sweep can drastically change the aerodynamic properties of the wing. The main idea behind sweeping wings is to decrease the effects of compressibility. This is because highly swept wings effectively decrease the mach number that the wing experiences. This directly influences the strength and the position of the shock wave over the airfoil, and thus the sweep can directly affect the total lift and drag that the wing produces. The next set of parameters that were selected as design variables were wing semi-span ($b/2$), root chord length (C_{root}), and taper (λ). Note that taper cannot be shown explicitly in Figure 4-3 but is still used as a design variable. Taper is defined by,

$$\lambda = C_{tip} / C_{root} \tag{4-2}$$

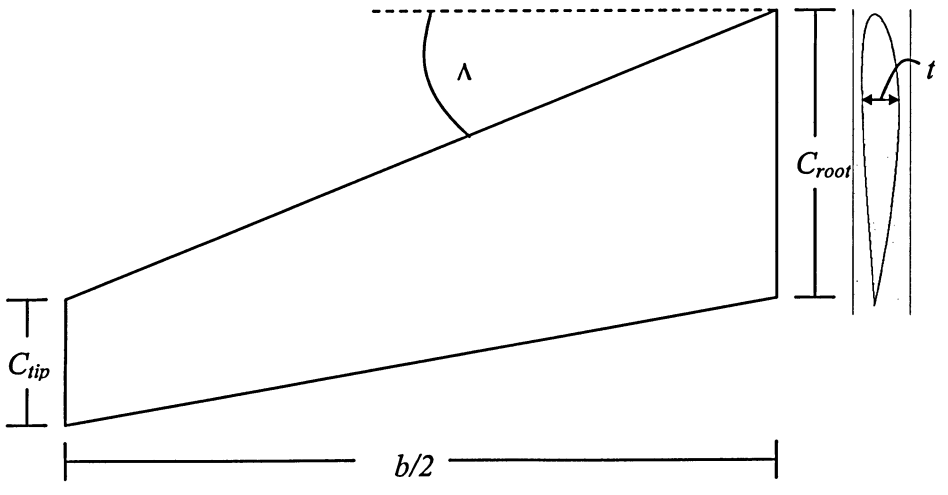


Figure 4-3: Wing model parametric definitions

where C_{tip} is the tip chord length. These three design variables, semi-span, root chord, and taper, when combined define valuable aerodynamic terms. The first has already been defined as taper, the second is aspect ratio (AR). Aspect ratio is defined by,

$$AR = b^2/S \tag{4-3}$$

where b is the span of the aircraft wings and S is the reference area of the wings. The wing reference area for one wing can be calculated by,

$$S/2 = \left(b/2\right)C_{root} - \left(\frac{C_{root} - \lambda C_{root}}{2}\right)\left(b/2\right) \tag{4-4}$$

Both taper and aspect ratio can drastically affect the aerodynamics of the wing. High taper ratios force more aerodynamic loading near the wing tip and thus increases the amount of wing deformation and wing root stress. Small taper ratios benefit aircraft design by decreasing the effect of the wing tip vortices and thus drag. Overall, balance must be maintained between the amount of lift required and the amount of drag created. Aspect ratio has a similar affect that taper ratio has on the wing. A high aspect ratio wing (long and slender) tends to have less drag for a given lift coefficient than low aspect ratio wings (short and stubby) do. This is due to the wing tip vortices and the distance the tip is from the fuselage. It may look like it is beneficial to have a wing that maximizes the aspect ratio. However, when the structural side of the design is considered it can quickly

be seen that a high aspect ratio wing can have high root stresses and thus will increase the overall aircraft weight. The final aerodynamic design variable is the overall airfoil thickness, *t*. Airfoil thickness can affect the drag, lift, stall characteristics, and structural weight. For design optimization airfoil thickness must be considered because a thinner airfoil will decrease the lift and it will decrease drag as well. More importantly statistical information has shown that the wing structural weight varies approximately inversely with the square root of the thickness/chord ratio [2]. There are many more intricate interactions between all of the design variables which the optimizer can take into account. The optimizer has the job of finding the optimum values of each, to maximize the performance.

4.1.3 Consistent Wing Geometry

The wing geometry that is defined in the above section must be created in two separate software packages. Because each software package requires different modeling techniques to create the wing there is the possibility that one model may be slightly different than the other. If this happens then there will be error introduced into the optimization. Therefore, it is necessary that these geometries be consistent between the CFD and FEM models. To make sure that both the CFD and FEM geometries are the same, a master geometry database was created that holds geometrical design variables for both the CFD and FEM. This however does not guarantee that the two models will be consistent. In order to maintain consistent geometry, ANSYS was used to create the original model. The ANSYS model is then exported to GAMBIT via an in house MATLAB function, for the creation and meshing of the CFD model. This way, the CFD model is based entirely on the geometry from the FEM model. This method of model generation guarantees the same geometries and removes any possible error between the two models. Both of the models must be able to provide consistent variant geometries that can be automatically generated without human interference [26, 61, 62]. It is especially important that all geometries must be able to be accurately generated within the design space. Any small inconsistency between the two models or an invalid geometry can introduce a degeneracy that can cause the optimization to fail.

4.2 Computational Fluid Dynamics (CFD)

The CFD model was created in GAMBIT and solved using FLUENT. These software packages were used because they were independent to the structural solver and thus forcing external coupling codes. This was required so that the future implementation of the IDF and CO MDO architectures was easier as they require decoupled solvers. The following section details the design criteria that went into making the CFD model.

4.2.1 Grid Generation

There are two main types of elements that can be used to create the solution domain. The first is the unstructured type that is generally used to conform to complicated shapes. The second is the structured type that is very popular in MDO wing design because it can easily conform to the wing contours [44, 49, 63, 64]. For the CFD model used in this work the structured grid was used because of its ease of implementation. The far-field was designed using a C-grid because of its inherent ability to keep grid skewness to a minimum. The far-field extremities were set to the dimensions that can be seen in Figure 4-4, where *c* is the airfoil chord length.

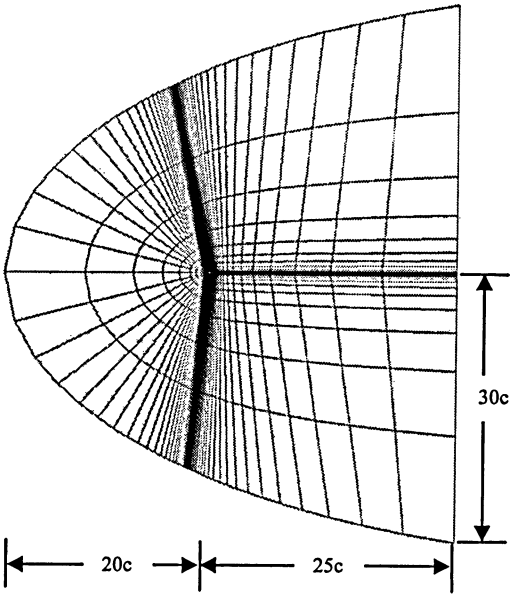


Figure 4-4: CFD far-field C-grid

The far-field dimensions were used because they minimized the skewness of the grid near the airfoil surface. The C-grid was extruded by 5 span lengths along the wing span axis to form a three dimensional far-field. An example of the 3-D grid used in this analysis can be seen in Figure 4-5. Note that the grid used for this analysis is rather coarse

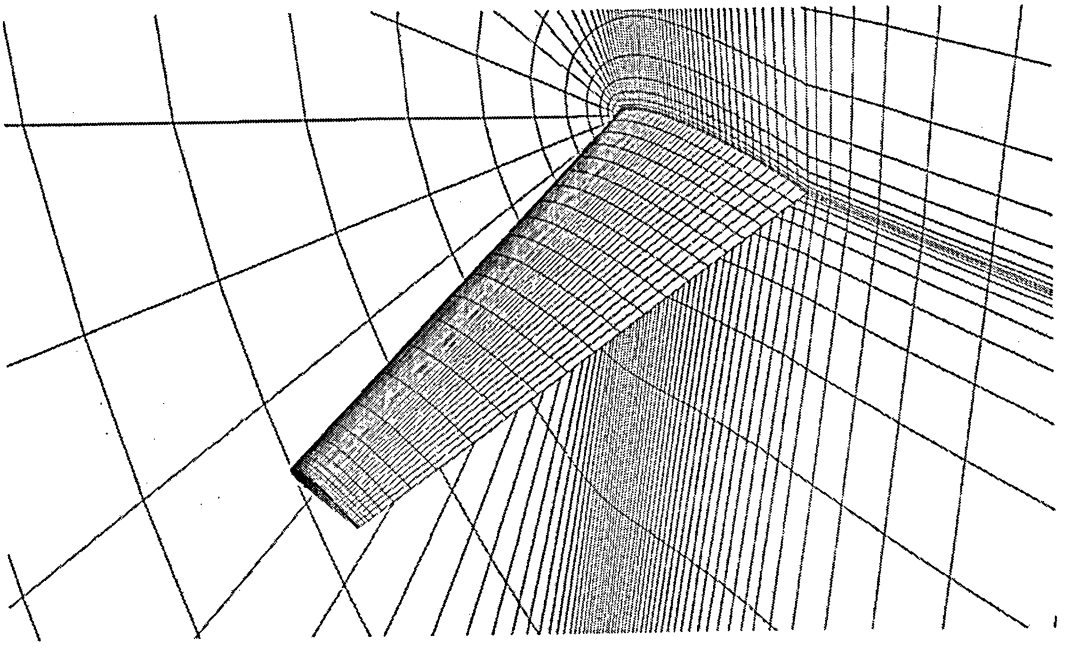


Figure 4-5: Example of wing CFD model

when compared to accepted CFD grids. This type of grid coarseness is acceptable for MDO and is done extensively in nearly all high-fidelity optimization research [44]. Increasing the coarseness of the grid helps to improve the solution time for the MDO because there are fewer equations to solve. The coarseness of the model must be a balance between the amount of information that is required and amount of time that is needed to obtain the information.

The CFD domain is solved using Euler equations instead of using the full Navier-Stokes equations to save computational time. In addition, the Euler equations were also used because they tended to decrease the amount of numerical noise that the solution returned [65]. Numerical noise is a problem that is inherent in all CFD solvers which introduce small amplitude high frequency noise into the numerical solutions. This numerical noise can cause problems when using finite differencing to calculate the gradients of the design variables. For example, Figure 4-6 shows two curves of a similar

trend, one is smooth and would allow accurate derivative calculations. The second curve contains noise and could introduce error into the derivative calculations.

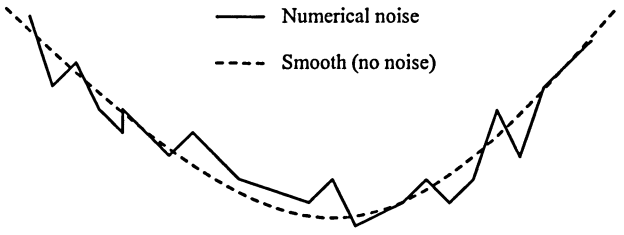


Figure 4-6: Example of numerical noise

4.2.1.1 Grid Skewness

The grid near the surface of the wing can often become highly skewed. This is undesirable because the numerical solver would prefer to have perfectly square elements in the solving domain. Having highly skewed elements can introduce numerical error and can even cause the solution to diverge. During the design of the CFD far-field, all attempts were made to minimize the grid skewness. An example of what type of grid is preferred is given in Figure 4-7. To minimize the grid skewness a built in function available in GAMBIT was used when designing the far-field. Skewness can also be

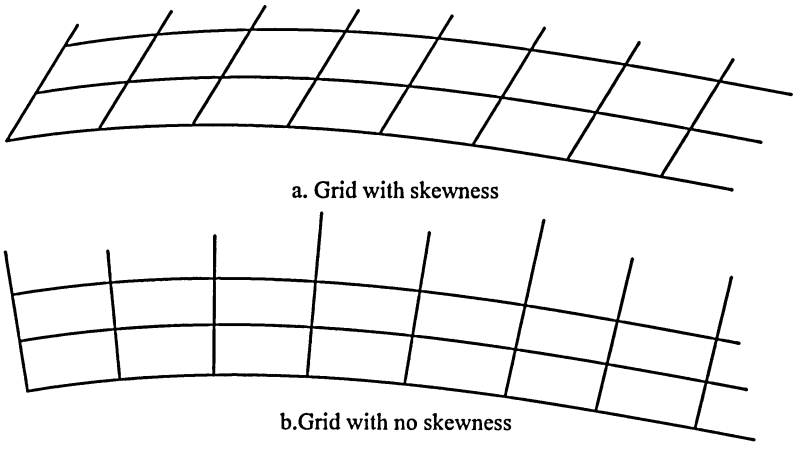
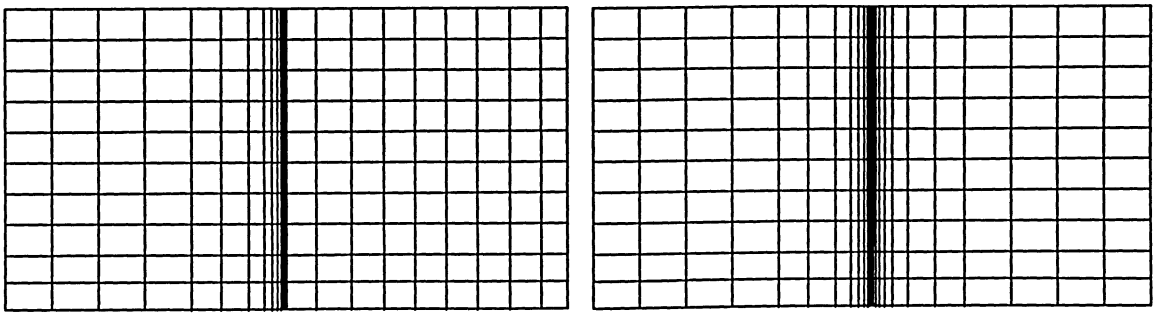


Figure 4-7: Example of grid skewness

introduced when sweeping the wing. Because the software that was created can mesh any wing shape the skewness due to wing sweep could not be minimized. This is because the automatic mesher uses a generic meshing routine to mesh all geometries.

4.2.1.2 Grid Continuity

Another grid design consideration is to maintain grid continuity over the entire solution domain. Often nodes are gathered together in order to accurately capture the changes in the flow and then they are spread out again when the changes are not expect to be large. Grids that contain discontinuous properties, as seen in Figure 4-8 a., can often induce numerical error into the solution because of the jump from one node spacing ratio to the next. A grid that looks like Figure 4-8 b. is more desirable because it minimizes the amount of error that can occur. Discontinuous grid properties can often occur when creating the mesh along the wing tips and the leading/trailing edges.



a. grid with discontinuous properties

b. grid with continuous properties

Figure 4-8: Example of grid continuity

4.3 Finite Element Method (FEM)

The finite element (FE) model that is used to calculate the structural response of the wing was created in ANSYS. ANSYS was used because the software package was available, and it contained the ability for implementation of an input file to help in the automatic wing generation. This section details how the automatic generation of the model is done and what types of elements are used to solve the problem.

4.3.1 Model Creation

The structural finite element model was created to contain the most basic elements of the airframe structure and to maintain enough design freedom to allow the optimizer to accurately obtain a global optimum. To give the most amount of freedom the wing was created from five main structural elements. Upper and lower skins were selected because they make up a large portion of the structural weight. In addition, spars, ribs, and stringers were also selected to make up the wing structure.

The first step in the modeling process was to determine where the spars were to be placed relative to the local chord. It was found that the leading edge spar is usually found somewhere between 10%-15% of the local chord, and the trailing edge spar was found to be placed approximately at 70%-80% of the local chord [66]. It was chosen that the spars for this study were to be placed at 10% and 75% of the local chord to be close to the values for the accepted spar placement. The next step was to determine the number of stringers and ribs that were to be used for the model. The number of ribs that was used for the model was chosen to be thirty because it was similar to the number of ribs for the commercial transport wing that Anhalt et al. [67] analyzed. The ribs were oriented to be parallel with the airflow because it made the automatic model generation easier. However, it is more common to orient the ribs so that they are perpendicular to the spars to help decrease manufacturing problems and to help decrease wing weight. The number of stringers was selected to be ten, which is typical for some commercial aircraft. The number of ribs and stringers was fixed to make the optimization easier. The structural model is shown in Figure 4-9 with only some of the ribs and stringers shown to make the diagram clearer. The upper skin has been removed to shown the internal structural elements such as the ribs, spars and stringers. A second figure shows the wing box without the skin and stringers.

The model was created by first creating the entire volume of the wing by simply extruding the airfoil shape along a direction defined by the sweep and span. The entire volume is then divided into many smaller volumes that will be later on used to identify the locations of the ribs and stringers. All of the volumes were then deleted but the areas were kept to result in a hollow wing. The next step is to identify which of the areas that

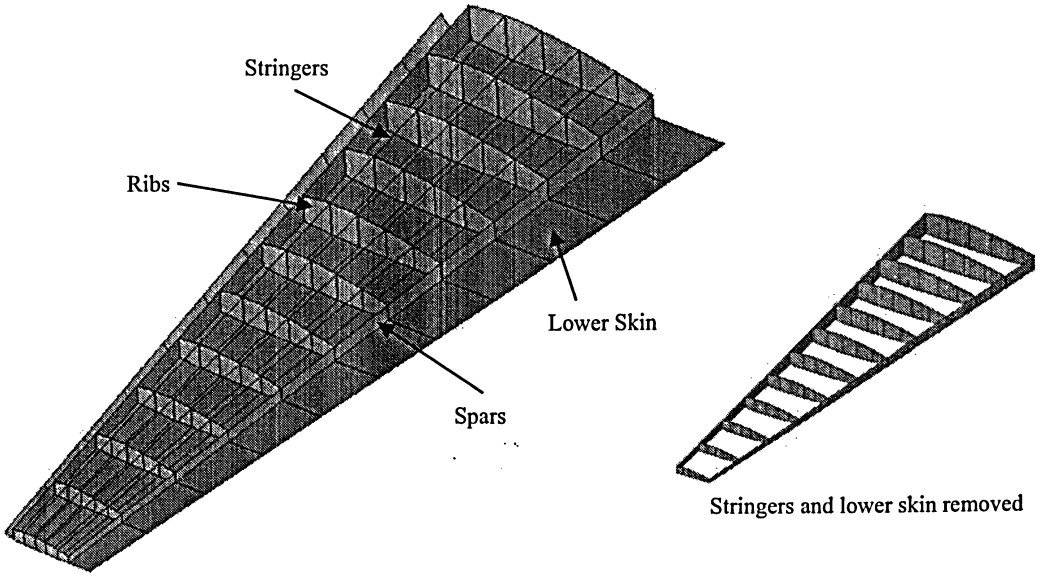


Figure 4-9: Structural elements within the wing

are remaining should be used for the meshing of the wing. This is easy when the GUI is being used in ANSYS but is extremely difficult when input files are used. This is because the selecting of the specific entity must be done by using what ANSYS refers to as *hot spots*. These *hot spots* are used to select entities located by a set of coordinates. A *hot spot* is what identifies where the center of the entity is. For example, an area contains a *hot spot* as defined in Figure 4-10 which needs to be located within a specific tolerance set within ANSYS. Locating these hot spots can be difficult because exact calculations of their coordinates can be hard to get. This is especially evident when trying to locate lines based on the line *hot spot*.

4.3.2 Mesh Generation

The structures that make up the shape of the wing are curved areas and straight lines. In order to accurately model the 3D wing properties appropriate 3D finite elements need to be chosen. Originally plate elements were used to model the curved areas, but it was found they were unable to fit the curved wing surface. Shell elements were chosen for the curved areas because they can take into account the warping of the element. They

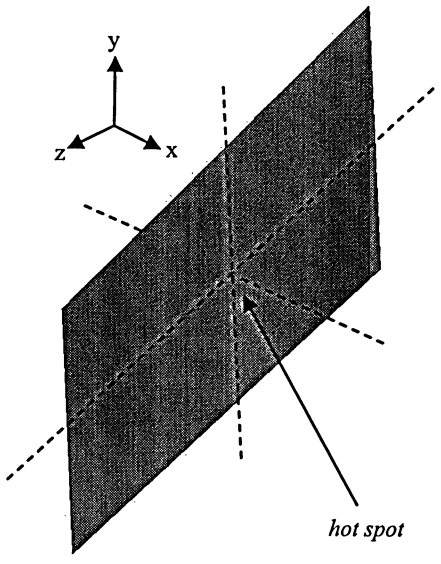


Figure 4-10: Location of a *hot spot* for an area

can also take into effect bending, shear, and torsional effects of the wing as well as in-plane or normal loading. The shells were used to model the ribs, spars, and wing skins. In the ANSYS environment the shell element used was '*shell63*'. This specific element was chosen because it is a linear elastic 3D finite element with 6 degrees of freedom at each node. The stress for this type of element can calculated for the shell mid-plane, the top surface, or the bottom surface. All stresses that are exported during the analysis are from the shell mid-plane. The straight lines, or stringers, were modeled using 3D beam elements to take into affect the bending, shear, and torsional effects of the wing. They were also selected because they are directly compatible with the 3D shell elements. Within the ANSYS environment the beam element '*beam4*' was chosen because it is also a linear elastic 3D finite element. Both of the elements can be seen in Figure 4-11.

The wing was meshed using a mapped hex mesh over the entire wing area which can be seen in Figure 4-12. The mesh quality was quite good, as the strain energy based error estimation given by ANSYS revealed that 95% of the elements have less than 3.57% error per element. The mesh density shown here was used because it provided enough nodes for the CFD grid points to be mapped to. The wing in Figure 4-12 only

shows ten of the wing ribs and none of the stringers. The root of the wing was constrained using a clamped boundary condition [68].

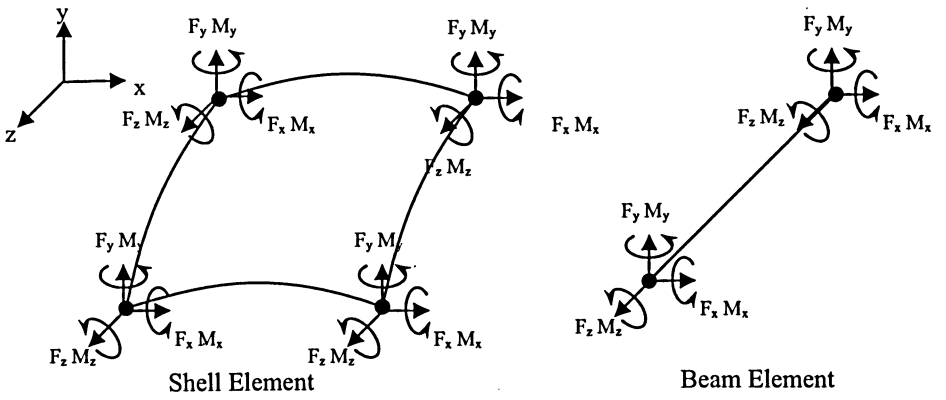


Figure 4-11: Finite elements used to model wing

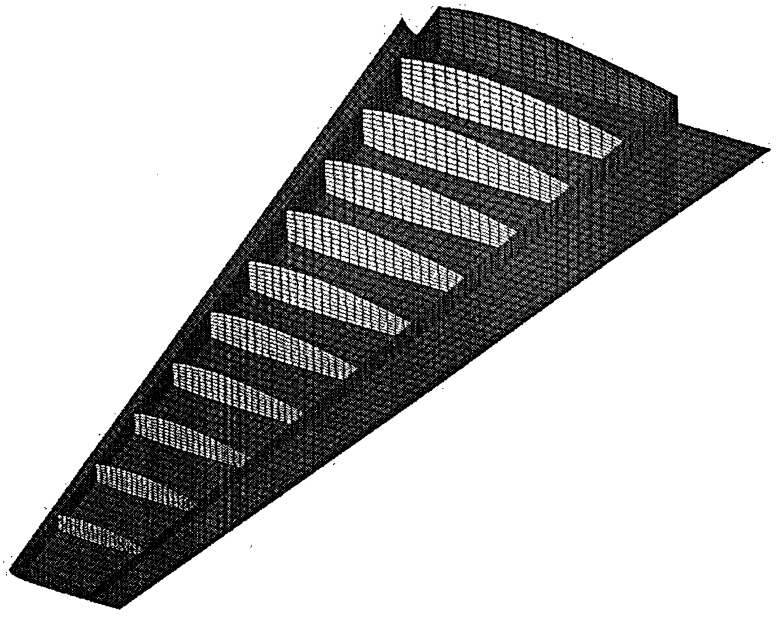


Figure 4-12: FEM mesh of the structural elements

4.3.3 Structural Taper

Each of the structural elements in the model has been structurally tapered to help decrease the overall wing mass and to help have an even stress distribution from the wing tip to the wing root. The structural taper was implemented in a piece-wise linear fashion with each jump in structural thickness occurring at each rib. The amount of structural taper is chosen by the optimizer as a percentage of the thicknesses that are at the root of the wing. For example, the Airbus A380 has a skin thickness at the wing root of 25 mm and a skin thickness at the wing tip of 5 mm [69].

4.4 Aero-Structural Coupling

The aero-structural coupling is required in order to determine the effect of wing flexibility on the optimization process. Computational aeroelastic problems such as the one described here can be solved using the strongly coupled approach or the loosely coupled approach. Loosely coupled aeroelastic problems as described by Bhardwaj et al. [70] are those that are solved in a modular format. This means that the CFD and FEM portions are solved independently and are integrated via a coupling code. The strongly coupled approach solves the CFD and FEM equations simultaneously where all the coupling is done internally. As mentioned previously the former method is taken with this research in order to be able to easily decouple the aero-structural interaction for use in the decoupled MDO architectures.

The static aeroelastic coupling procedure was outlined in Chapter 3. This chapter contains a much more detailed description of how the coupling between the two disciplines is achieved. The entire process can be seen in Figure 4-13. The process

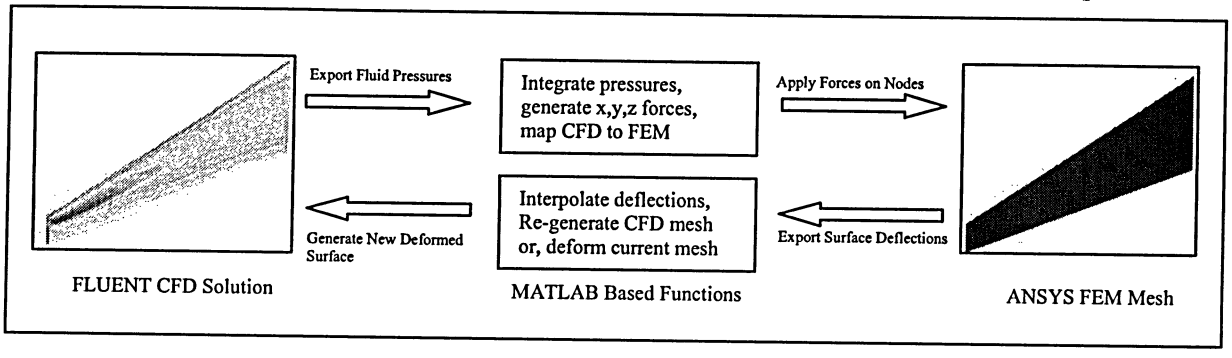


Figure 4-13: Static Aeroelastic coupling procedure

begins by solving the CFD solution which can be done using any CFD software. Once the solution is obtained the pressures are exported at desired intervals along the span of the wing. The pressures are taken and integrated along the span to determine the normal forces on the wing. Based on the current wing deformation, the x, y, and z forces are then generated. Details concerning this process can be found in Sec. 4.4.1. The next step is to map the CFD grid to the FEM mesh in order to accurately transfer the calculated forces to the corresponding FEM node. Once this is done, the surface deflections from the FEM structural response are exported and used to regenerate the CFD mesh. The new CFD configuration is solved and the solving procedure is repeated until the convergence criteria are met. For this work, the static aeroelastic convergence is obtained when neither the trailing edge tip deflection nor the tip twist vary by 1.0^{-4} from one iteration to the next.

Convergence of the aeroelastic solution using the loosely coupled solving method is CPU intensive. There are methods that can be implemented into the solving procedure that will speed up the rate of convergence of the aeroelastic problem. The first method is to introduce an additional mass [71] to the wing, which increases the wing inertia and damps the static aeroelastic convergence. Adding more mass can be done in a number of ways; the method that is used for this work, as an example, is a point mass was attached to the wing to simulate an engine and nacelle. The next method to help improve convergence speed was to use under relaxation techniques that were mentioned previously [47,72,73]. The final method is to use only intermediate CFD solutions. This way time can be saved by quickly obtaining a configuration that is close to the steady-state solution [35,74]. The only stipulation on this method is that the CFD solution must be fully converged only on the final aeroelastic iteration.

Sometimes the aeroelastic coupling will not converge because the free-stream velocity is above the divergence velocity. However, this is typically not the case when dealing with swept wings because they have high divergence speeds [59]. For this research, divergence was never found to occur because swept back wings were always used for static aeroelastic solution.

4.4.1 Load Transfer

Once the pressure distribution has been obtained from the CFD analysis the equivalent nodal forces must be found and applied to the finite element model. This is done using a similar process that was developed by Bhardwaj et al. [59, 70]. The process is generalized so the pressure distribution can be found using any CFD code and used with this method as long as local geometrical information is available.

To begin, the FLUENT module is called via the MATLAB wrapper (see Appendix B for details). When the CFD solution has converged, the FLUENT module begins the exportation of the pressure distribution at user defined intervals. A study, which can be seen in Figure 4-14, was done to determine how many span wise pressure

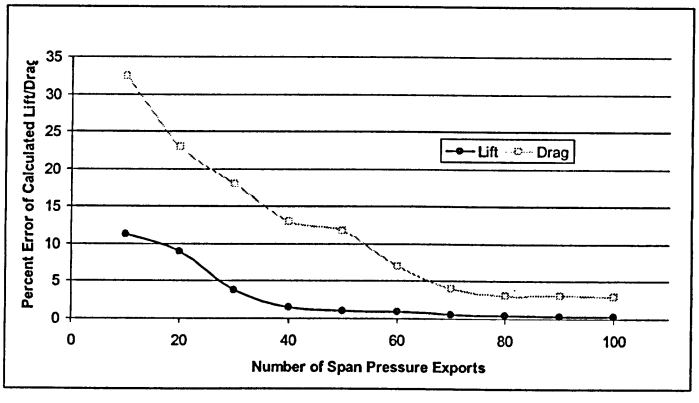


Figure 4-14: Percent error of calculated lift and drag versus FLUENT lift and drag

distributions that would be needed to accurately estimate the lift and drag forces on the wing. It was found that in order minimize the amount of error between the calculated lift/drag and the FLUENT lift/drag that at least 80 span wise pressure exports are needed. Exporting at least 80 pressures will allow the calculated lift and drag forces to be estimated to within 0.5% and 3% of the FLUENT lift and drag forces respectively. The percent differences are then used to calibrate nodal forces to match the lift and drag forces given directly from FLUENT to ensure an accurate load transfer. Note that the distribution of the force is not changed; only the total resultant of the force is scaled.

The first step in transferring the CFD pressures to FEM forces is to integrate the pressures in order to find the normal forces acting on the wing. All of the CFD grid points are exported and used to manually calculate the local area to be used for each pressure via,

$$S^{i,j} = \left(\sqrt{(x^i - x^{i+1})^2 + (y^i - y^{i+1})^2}\right) \cdot z^j \tag{4-5}$$

where S^{ij} is the local cell area, x is the local value of the x coordinate, y is the local value of the y coordinate, and z is the length of the local span wise interval. The superscripts i and j represent the chord wise and span wise indices respectively. Each of the locally calculated areas is then used to convert the pressures into a normal force on the wing via,

$$F_n^{i,j} = P^{i,j} \cdot S^{i,j} \tag{4-6}$$

where $F_n^{i,j}$ is the normal cell based force and P^{ij} is the cell centered pressure acting on the wing. The graphical representation of the definitions from equations (4-5) and (4-6) can be seen in Figure 4-15.

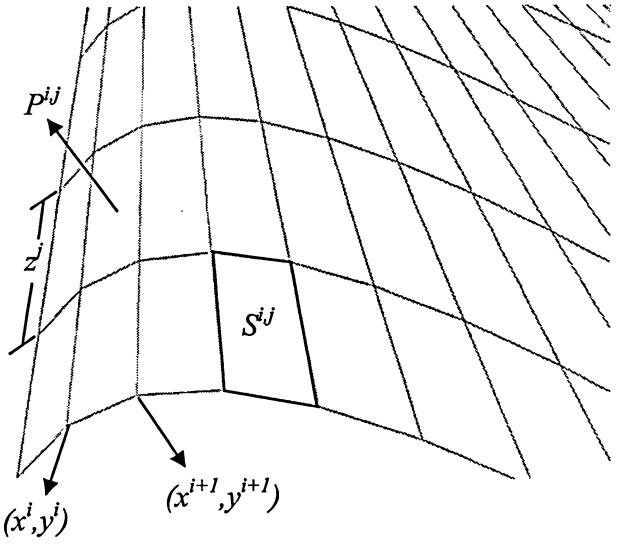


Figure 4-15: CFD grid area calculation

It is necessary to compute the aerodynamic forces based on the CFD wing boundary and geometries to maintain a conservative coupling interaction [58] as opposed to using FEM geometries. Now that the normal force acting on the cell has been calculated, the components in the x, y and z direction must be calculated for application to the FEM nodes. In order to do this, all of the surface unit normals must be calculated between the CFD grid points. This is done using the following equation.

$$\begin{Bmatrix} n_x^{i,j} \\ n_y^{i,j} \\ n_z^{i,j} \end{Bmatrix} = \frac{(x^i - x^{i+1})_x + (y^i - y^{i+1})_y + (z^j - z^{j-1})_z}{\sqrt{(x^i - x^{i+1})^2 + (y^i - y^{i+1})^2 + (z^j - z^{j-1})^2}} \tag{4-7}$$

All of the calculations for local grid area and grid unit normals are done in such a way that grid areas that exist on the leading or trailing edge, the wing root, or wing tip are calculated correctly. The normals are then corrected so the cosines are valid for the forces normal the wing surface. The overall force that acts on the local cell can be calculated by combining equations (4-5)-(4-7) to get,

$$\begin{Bmatrix} F_x^{i,j} \\ F_y^{i,j} \\ F_z^{i,j} \end{Bmatrix} = P^{i,j} S^{i,j} \begin{Bmatrix} n_x^{i,j} \\ n_y^{i,j} \\ n_z^{i,j} \end{Bmatrix} \tag{4-8}$$

Once all of the calculations have been completed for every single cell within the CFD grid the forces must be mapped onto the FEM mesh. The mapping that is described here is only effective if the CFD grid is more dense that FEM mesh. This is because it is necessary to have a majority of the FEM nodes loaded with forces in order to avoid any extremely high stress concentrations due to overloading of a node.

The first step in mapping the CFD grid to the FEM mesh is to split the grid and mesh points of the wing into the lower and upper surface. This makes the entire process less prone to errors because the CFD upper surface grid points are guaranteed to be mapped to the FEM upper surface nodes and vise versa. The mapping process that is used here is a simple algorithm that picks a CFD grid point and tries to find the nearest FEM node. This is done using,

$$d^{i,j} = \sqrt{\left(x_{cfd}^{i,j} - x_{fem}^{i,j}\right)^2 + \left(y_{cfd}^{i,j} - y_{fem}^{i,j}\right)^2 + \left(z_{cfd}^{i,j} - z_{fem}^{i,j}\right)^2} \tag{4-9}$$

where d^{ij} is the distance from the CFD grid point to the FEM mesh node, the point $\left(x_{cfd}^{i,j}, y_{cfd}^{i,j}, z_{cfd}^{i,j}\right)$ are the grid coordinates from the CFD grid point *i,j* and the point $\left(x_{fem}^{i,j}, y_{fem}^{i,j}, z_{fem}^{i,j}\right)$ is the FEM node that is being considered for force transfer. The algorithm stores all of this information and picks the nearest FEM node to be used for the force application. Examination of Figure 4-16 reveals that there could possibly be a problem when two FEM nodes are equidistant from a CFD grid point. This can be seen

when considering CFD grid point (i-1, j) and FEM nodes 1 and 3. When this happens the force is divided and applied equally to each node. It can also be seen here that some nodes will receive more forces than others. This was found not to be a problem because

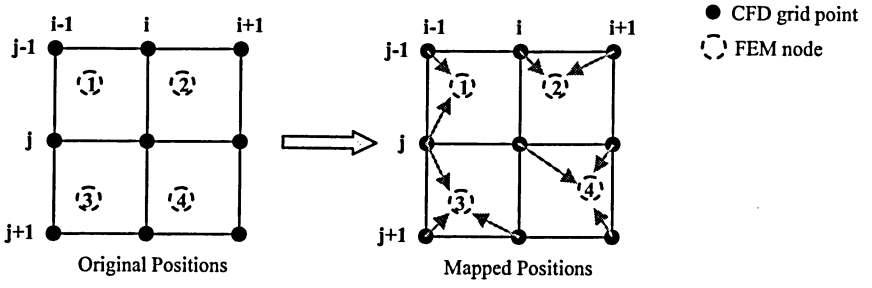


Figure 4-16: Mapping CFD grid points to FEM nodes

the distance between nodes is small and thus the overall effect of the moment that is created is small. When mapping the CFD grid to the FEM mesh it is absolutely necessary to use the original unloaded CFD wing shape when doing this. This is because the FEM mesh is always from the unloaded configuration. Using the updated CFD wing shape will result in inconsistent load transfer for each iteration and possible non-convergence of the optimization.

4.4.2 Displacement Transfer

After the loads are applied to FEM model and the structural response has been calculated, the deformations must be used to update the CFD grid. In order to simplify the displacement transfer procedure, it was assumed that each wing cross section remains the same shape. This means that even though small changes in the camber and wing thickness can occur it is so small that there is a negligible effect. Consequently, the deformation that occurs to the wing is limited to simply translation in the y direction and rotation [74] about span axis of the wing. Cavallo [75] has shown that small changes of the wing shape in the x and z directions have negligible effects on pressure distributions and thus can be neglected when updating the geometry. For this research it was found that the displacement that occurs in the span wise direction never rises above 1.0^{-3} which is deemed minimal and can be ignored. These assumptions make updating the CFD model easy because the new geometry is based entirely on vertical translations and changes in the twist angle.

Since the structural response is calculated using ANSYS, it is easy to export data that defines the wing deformation. The only two pieces of information needed to update the wing geometry for the CFD analysis are the leading edge and trailing edge lines. These are exported as a series of data points that are used to create a b-spline from the root to the tip. Since the shape of the airfoil is assumed to be constant, the new shape of the exterior wing surface is generated by using the leading and trailing edge b-splines as guides. From visual comparison of superimposed CFD and FEM models it was found that this type of displacement transformation was geometrically consistent. Using this method to update the CFD model, it is necessary to completely regenerate the CFD grid at each iteration. It is possible using FLUENT to deform the existing CFD grid but this would have added another level of complexity and may not have yielded better results.

Chapter 5

Aero-Structural Optimization Results

5.1 Sensitivity Analysis

There are several different methods that can be used to calculate the sensitivity gradients for the aero-structural optimization. The easiest and most straight forward method is finite differencing by Taylor's series expansion. For this work, forward finite differencing is used in general, and when the optimization is at the upper bound of a design variable backward finite differencing is used. The first derivative approximation can be obtained by using the Taylor's series expansion of a general function *f(x)*,

$$f(x+h) = f(x) + hf'(x) + h^2 \frac{f''(x)}{2!} + h^3 \frac{f'''(x)}{3!} + \ldots \tag{5-1}$$

the first derivative can be found by eliminating higher order terms and solving for *f'(x)*:

$$f'(x) = \frac{f(x+h) - f(x)}{h} \tag{5-2}$$

where $f'(x)$ is the first derivative of the function and h is the step size for the derivative. The truncation error due to the higher order terms is *O(h)*, which means that finite differencing is only first order accurate. New techniques to calculate sensitivity gradients such as the couple-adjoint sensitivity method [46] and global sensitivity equations [24] provide quicker means of obtaining all of the sensitivities compared to finite differencing. These advanced methods were not used for this work because their computational efficiency is generally only taken advantage of when the number of design variables

exceeds 18 [46, 76]. This is because these advanced sensitivity methods have been shown to be computationally independent of the number of design variables. Martins et al. [46] showed that using the coupled-adjoint method, sensitivities can be calculated for 200 design variables for the computational cost of 18.

An important aspect of this work was to determine the proper finite differencing step size to use when calculating the derivatives. This is important because extremely small finite difference step sizes will have negligible effects on the aerodynamics of the wing. For example, the difference in the wing L/D ratio is negligible when comparing a wing sweep of 30° to a wing sweep of 30.000001°. It was found, using a sensitivity analysis study, that the structural design variables are sensitive to changes up to a step size of 1.0^{-6} while the aerodynamic step sizes are sensitive only up to a step size of 1.0^{-4}. Thus, for the calculations of the derivatives during the optimization a step size of 1.0^{-4} was implemented. A larger step size would have revealed a flat design space with no optimum and a smaller step size would have caused inaccuracies in the derivative due to the finite differencing of the solution domain.

5.2 CFD Validation

Before the CFD analysis could be trusted to perform the aerodynamic calculations several test cases were completed. This was done to check that the geometry that is created by the optimizer could be used with confidence. In addition, the test cases were done to provide a better understanding of the software being used and to provide additional CFD experience. The first test case is for a simple 2D analysis and the second case is for 3D flow over a wing.

5.2.1 2D NACA Airfoil

There are two main reasons for running the 2D tests cases for the CFD solver. The first is to ensure that the NACA 4-digit airfoil creator was able to calculate accurate airfoils over a wide range of geometries. The second is to make certain that the CFD grid techniques being implemented were also accurate.

The CFD analysis was done using the Spalart-Allmaras viscous model at mach 0.15 with an angle of attack of 4°. All of the other initial conditions used for the analysis were set to sea-level conditions.

The CFD data obtained from FLUENT was benchmarked against data provided by Abbott et al. [77]. A summary of the calculated data is given in Table 5-1 seen below.

Table 5-1: 2D airfoil benchmarking results

NACA 0012			
	C_L	C_D	L/D
Geometry Maker	0.443	0.0118	37.542
Abbott	0.440	0.0115	38.261
Percent Difference	0.677	2.542	1.914
NACA 1410			
	C_L	C_D	L/D
Geometry Maker	0.557	0.0116	48.017
Abbott	0.530	0.0110	48.182
Percent Difference	4.847	5.172	0.343
NACA 2412			
	C_L	C_D	L/D
Geometry Maker	0.670	0.0128	52.344
Abbott	0.640	0.0114	56.140
Percent Difference	4.478	10.938	7.253
NACA 2418			
	C_L	C_D	L/D
Geometry Maker	0.637	0.0145	43.931
Abbott	0.640	0.0128	50.000
Percent Difference	0.471	11.724	13.815

Inspection of the results given in the table above shows that the airfoil maker used for the optimizer can accurately create a variety of airfoil shapes which can be used to yield an approximate solution of the airfoil properties. The percent differences are attributed to several things; the first being that the grid density used for the solution was rather course when compared to accepted CFD standards. The second possible reason for the difference between the data calculated using FLUENT and the Abbott data was that the Abbott data was read and interpolated from graphs where only specific data points were available, and some inaccuracies occurred when recording the data. It should be noted that as the airfoil gets thicker, as given by the last two digits of the NACA designation, the percent difference tends to increase. This is because for thicker airfoils the amount of skewness around the airfoil increases and results in more error. This is

avoidable when manually meshing because alterations within the grid can be made to minimize the skewness. However, the automatic mesher created for this research was used to automatically mesh each of the above airfoils used for this validation case and thus mesh alterations could not be made. To decrease the amount of error and numerical noise due to the grid skewness, the maximum thickness for the airfoil was limited during the optimization.

5.2.2 Onera M6 Wing

The 3D validation case was chosen to be the Onera M6 wing [78] because it has been studied extensively and is often used for validation of external flow CFD code. Many researchers have used the Onera M6 wing to perform aerodynamic optimization by changing the airfoils shapes and holding the wing plan form shape constant [53,79,80]. The Onera M6 is a popular comparison because of the abundant data that is available for the baseline case. The Onera M6 has the following geometrical properties as seen in Figure 5-1.

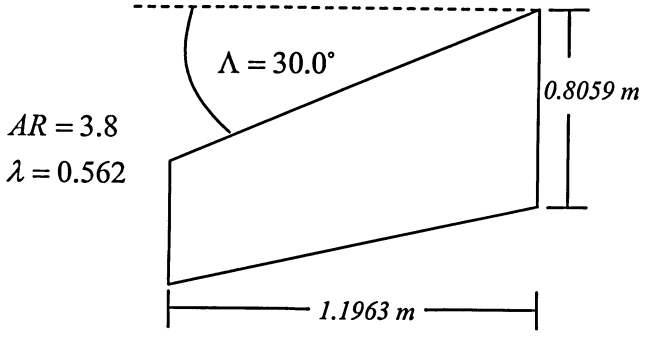


Figure 5-1: Onera M6 wing geometry

The Onera M6 wing validation case was solved using an angle of attack of 3.06°, mach number of 0.84, temperature of 255.4°K, pressure of 31.6 KPa, with a Reynolds number of 11.7×10^6. It can be seen in Figure 5-2 that two pressure distributions along the span of the wing compare well against the accepted experimental results [78]. The differences in the locations of the shockwaves are attributed to the coarse mesh density that is used to mesh the wing which cannot accurately predict the shock location. Finding the exact location of a shockwave can be a difficult task even for experienced CFD user. This was shown by Girodroux-Lavigne et al. [68] where several aerospace companies compared

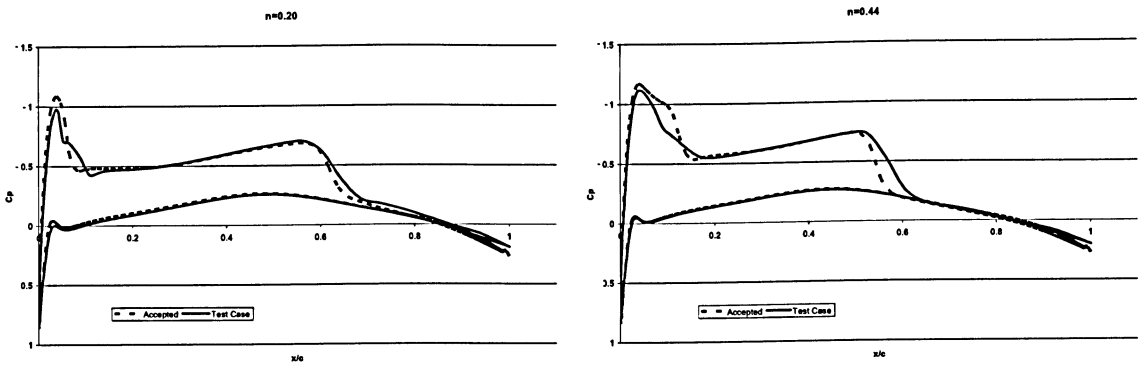


Figure 5-2: Onera M6 wing validation

CFD results. It was found that most of the data was consistent when comparing pressure distributions except for the strength and locations of the shockwaves.

The previous validation case has been shown to perform well when compared to the accepted validation results [78]. Thus, it is concluded that the geometry creation and grid techniques that are used by the optimizer can be trusted to generate accurate wings that can capture small details in the flow.

5.3 Static Aeroelastic Convergence Study

The computational static aeroelastic analysis that was done for this optimization is the most important aspect of this research. If the analysis does not perform correctly then the sensitivity gradients that are calculated by the optimizer will not be able to find a search direction gradient. To ensure that the static aeroelastic analysis was performing correctly a number of convergence tests were done. In addition, a test case was run to determine if the deflections and rotations of the wing were in the appropriate range when compared to other flexible wing models.

All of the convergence test cases and deflection/rotation results were solved for an arbitrary wing configuration that is given below in Table 5-2 and Table 5-3. These configurations have no relevance to any optimum design, they were chosen strictly because they would display noticeable wing deformations.

Table 5-2: Aerodynamic configuration for static-aeroelastic test cases

Sweep (deg)	Taper	Croot (m)	Semi-span (m)	t/c
35	0.3	12	27.5	0.1

Table 5-3: Structural configuration for static-aeroelastic test cases

Upper Skin (mm)	Lower Skin (mm)	Spar (mm)	Rib (mm)	Stringer Height (mm)	Stringer Width (mm)	Structural Taper
19	14	18	15	10	10	0.3

The wing deflection and rotation based on the above wing configuration can been seen in Figure 5-3. The displaced wing configuration is shown in blue and the original

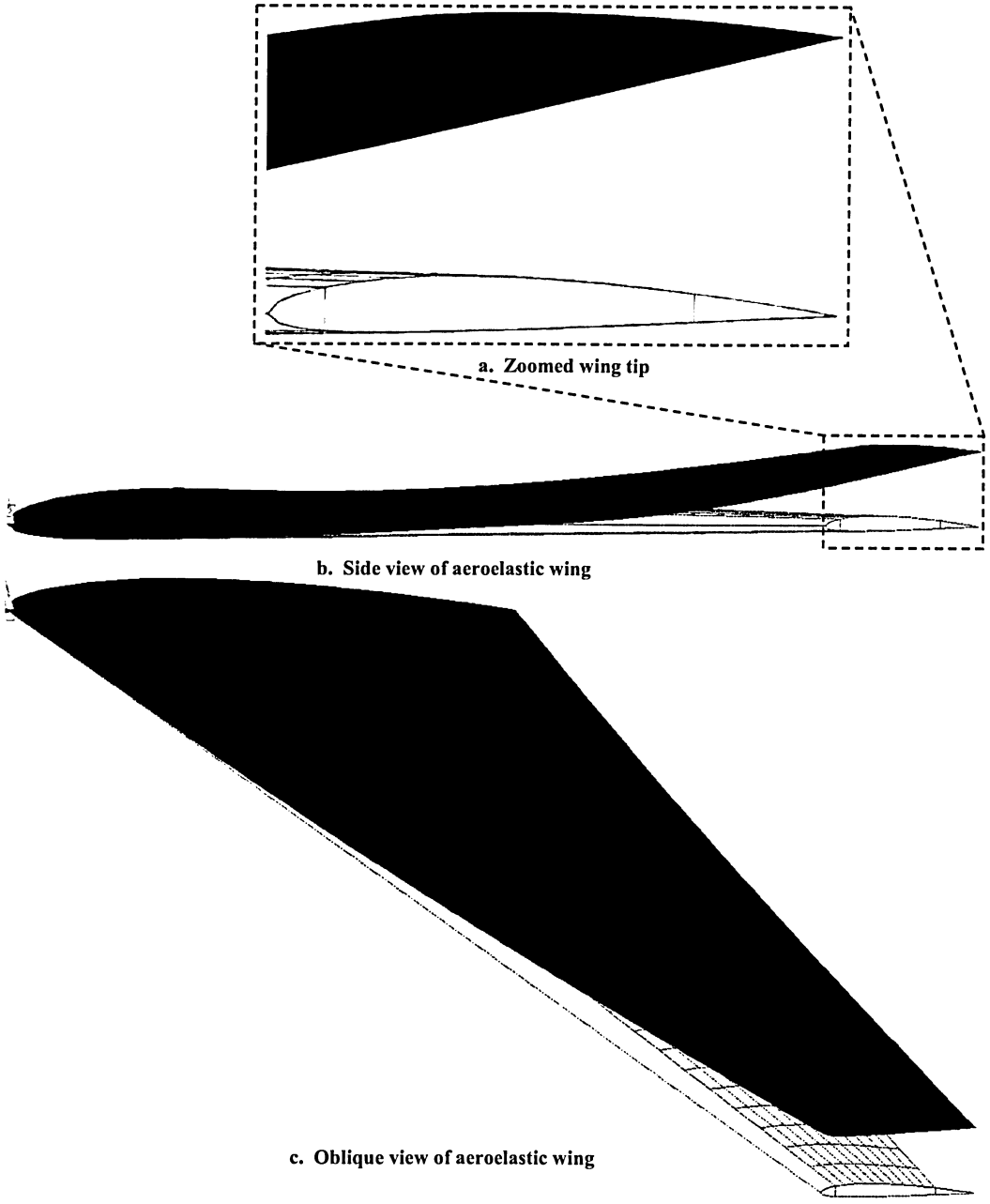


Figure 5-3: Aeroelastic deflection of an arbitrary wing configuration

wing (rigid wing) is shown outlined in black. The pitch angle for this case was calculated to be -1.94° and the wing tip deflection was 1.39 m. When compared against available data it was found that these values for pitch angle and deflection were in the appropriate range for a wing this size [68, 81]. The negative pitch angle is expected because for swept back wings the aeroelastic axis is located in front of the quarter chord line [46]. It was found that the maximum wing deflection, in all wing configurations during the optimization, always occurred at the trailing edge wing tip.

Three separate static aeroelastic convergence tests were run using the above wing configuration. The convergence plots of these tests can be found in Figure 5-4. The first test case was a baseline test that used a fully converged CFD solution to calculate the structural response of the wing. The wing converged to a static position within 10 static aeroelastic iterations which compares well with results obtained by Alonso et al. [47] as well as the generalized data given by Zeiler [82].

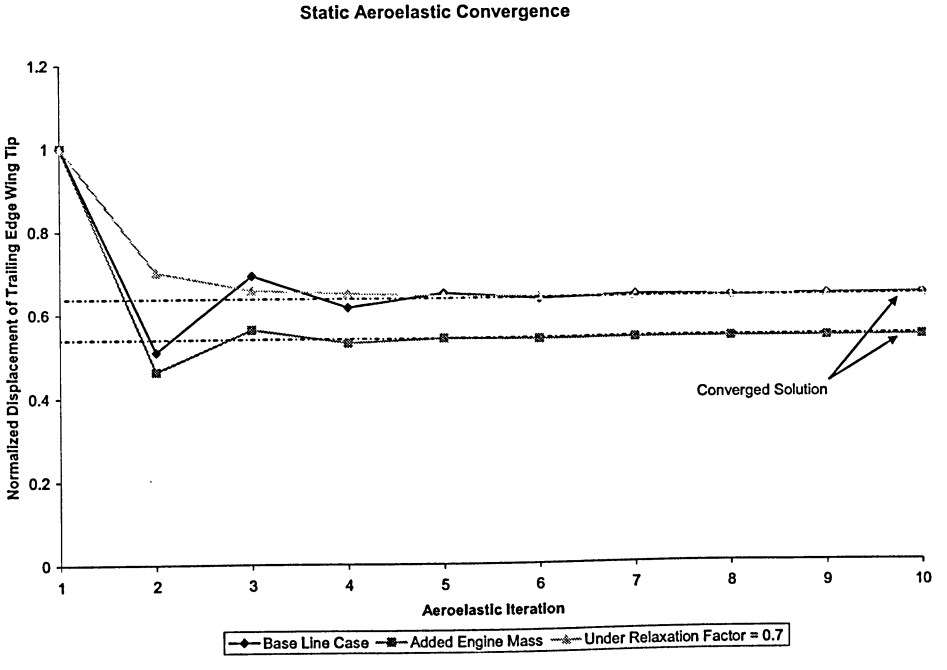


Figure 5-4: Static aeroelastic convergence study results

The second test case was completed to make sure that the wing would respond as expected with additional mass added to the wing. The added mass was a point mass attached to the wing by rigid multi-point constraint (MPC) structural elements. This configuration was setup such that the point mass acted as an engine attached to the wing.

The addition of the engine mass caused the static aeroelastic solution to converge faster than the base line case because of the added inertia to the system. This configuration converged to a different normalized displacement because of the increased mass. The last test case was the implementation of the under relaxation factor. Using an under relaxation factor of 0.7 it was found that the wing converged to a static position much quicker compared to no under relaxation factor. It can also be seen that the convergence occurred with no oscillations in the static aeroelastic position. This result was also expected and it compared well against results given by Alonso et al. [47] and Giunta [76]. Several other test cases were run where non-converged CFD solutions were used to obtain the structural responses. It was found that increasing the number of CFD iterations on each static aeroelastic iteration increased the number of static aeroelastic iterations required to obtain convergence by 2 iterations. However, the benefits of using the non-converged CFD solutions were seen when comparing wall clock times of the static aeroelastic convergence. The non-converged CFD solution method obtained a static wing response in about 80% of the time required for fully converged CFD solutions.

The convergence rates that are reported here are for the test wing configuration given previously. More flexible wings will take longer to converge to a static aeroelastic response, whereas more rigid wings will take a shorter amount of time. For the flexible wing optimization the non-converged CFD solution method and under relaxation method were combined to help speed up the solution process.

5.4 Optimization Setup

The optimization described here is only for a single wing of a commercial transport. To enable the use of the Breguet range equation data was taken based on standard commercial transports to give estimates of the empty aircraft mass and fuel capacity. The base weight of the aircraft with no wings was chosen to be $W_{Fuselage}$=127,000 kg based on statistical data and the mass of the fuel was set to W_F=81,590 kg [83]. During the optimization the fuel mass is scaled based on the total wing volume so that a wing larger than a Boeing 777 will have more fuel and a smaller

wing will contain less fuel. Based on the size of the wing, the optimizer can calculate the total weight of the aircraft via,

$$W_T = W_{Fueselage} + 2W_W + W_F \tag{5-3}$$

where W_T is the total mass of the aircraft, and W_W is the mass of the wing calculated by the optimizer.

The wing optimization was done for the conditions at 9,144 meters (30,000 feet) for the cruise setting. The airspeed used for analysis was set to mach 0.8 based on altitude conditions and the angle of attack was set to 1°. The airfoil shape was chosen to be the NACA 24XX, where XX is the optimizer calculated airfoil thickness.

5.4.1 Objective Function

The goal of the optimization is to maximize the overall range of the aircraft using the Breguet range equation. The Breguet range equation was used because it provides a method of combining the aerodynamics and structures disciplines as seen below,

$$R = k\frac{C_L}{C_D}\ln\left(\frac{W_T}{W_T - W_F}\right) \tag{5-4}$$

where C_L and C_D are the coefficients of lift and drag respectively, W_T is the total aircraft mass, and W_F is the mass of the fuel. The coefficient k is defined by V_∞/SFC where V_∞ is the free stream velocity; which is held constant during the optimization. The coefficient k, is used in lieu of specific data on the specific fuel consumption (SFC) of the engines and was arbitrarily chosen to be 800. Using this equation it can be seen that the best way to improve the range of the aircraft is to maximize the C_L/C_D ratio and to minimize the overall mass of the aircraft. However, the Breguet range equation cannot be used alone as there are several constraints that cannot be in violation upon optimizer convergence.

5.4.2 Constraints

There are several constraints that are used during the optimization, the majority of which are strictly used for design variable upper and lower bounds. The bounds were difficult to choose because if the bounds are too large it may take a long to time for the optimizer to converge. If the bounds are too restrictive a feasible design may not be possible in the designated design space. After completing many trial optimizations it was

determined that the upper and lower bounds given in Table 5-4 and Table 5-5 would give the optimizer lots of freedom in selecting aircraft configurations while not drastically increasing the optimization convergence speed.

Table 5-4: Aerodynamic design variable bounds

	Sweep (deg)	Taper	Croot (m)	Semi-span (m)	t/c
Lower Bound	0	0.3	6	20	0.1
Upper Bound	35	1	13	35	0.16

Table 5-5: Structural design variable bounds

	Upper Skin (mm)	Lower Skin (mm)	Spar (mm)	Rib (mm)	Stringer Height (mm)	Stringer Width (mm)	Structural Taper
Lower Bound	6	6	6	6	5	5	0.1
Upper Bound	30	30	40	40	20	20	0.35

The next constraint was an inequality constraint to make sure that the stresses within the structure are never above a specified limit. The material that is used during the optimization is aircraft grade 7075-T6 aluminum alloy with properties shown in Table 5-6 [84].

Table 5-6: Structural material properties of Aluminum

Density (kg/m^3)	Poisson	Yield Strength (MPa)	Max Tensile Strength (MPa)
2710	0.33	500	570

For the stress constraint the yield strength of 500 MPa cannot be used directly because there are a few F.A.A. regulations that must be implemented. The first regulation from section 25.305 [85] is that an aircraft structure must include a factor of safety of 1.5 for commercial transports. The second regulation is that the structure must be able to withstand the limit loads without any detrimental permanent deformation. The limit load is calculated via the load factor, n, which is usually calculated using a V-n diagram based on the aircraft size and weight. For the optimization done here, a single V-n diagram cannot be used because the wing size and weight change from one iteration to the next. It would have been possible to create a new V-n diagram on every iteration but it was determined that would add another level of complexity to the optimization, and it would

not necessarily improve the validity of the results. For illustrative purposes of the optimization it was chosen that a load factor of 3 because it would provide an acceptable wing design regardless of the wing size and weight. Applying the load factor and the factor of safety to the material yield strength the following constraint can be made:

$$\sigma_{\max} \leq 111 \ \ MPa \tag{5-5}$$

where σ_{max} is the maximum Von-Mises stress taken from the finite element model. The maximum stress is found by exporting all of the stresses from the finite element model and finding the maximum. The Von-Mises failure criterion was used because it provides a good approximation of all of the combined stresses acting on the system.

The final constraint is used to make sure that when the final aircraft configuration is optimized that the aircraft lift will be the same as the aircraft weight. The constraint can be written as,

$$L = W_T \tag{5-6}$$

where L is the lift generated by two wings, and W_T is the total aircraft weight. This constraint is implemented because the wing is designed for the cruise condition of the aircraft.

5.5 Wing Optimization Results

5.5.1 Rigid Wing Results

The optimization of the rigid wing was completed using five aerodynamic design variables and seven structural design variables. The large computational expense in this type of optimization is entirely due to the calculations of the derivatives of all of the design variables with respect to the objective function and the constraints. To help speed up the optimization, a database of all the pressure distributions was created so some pressures could be loaded from a file instead of calculated again using the CFD software. This was helpful when calculating the derivatives for the structural design variables because for the rigid wing optimization only the aerodynamic design variables can change the pressure distribution. When calculating the structural derivatives, pressures were loaded from the database and applied to the new structural model instead of re-

calculating the pressure distribution using CFD. Using this method the solving time for the rigid wing case was decreased by over half.

Several test cases were run to determine an appropriate starting point for the optimization that would be able to find the global optimum. A number of trials were run to see if the different starting points would find the same optimum. A comparison of the original and optimized wing plan form shapes can be seen in Figure 5-5. In addition, a completed list of the optimal design variables that corresponds to the optimized wing configuration is displayed in Table 5-7.

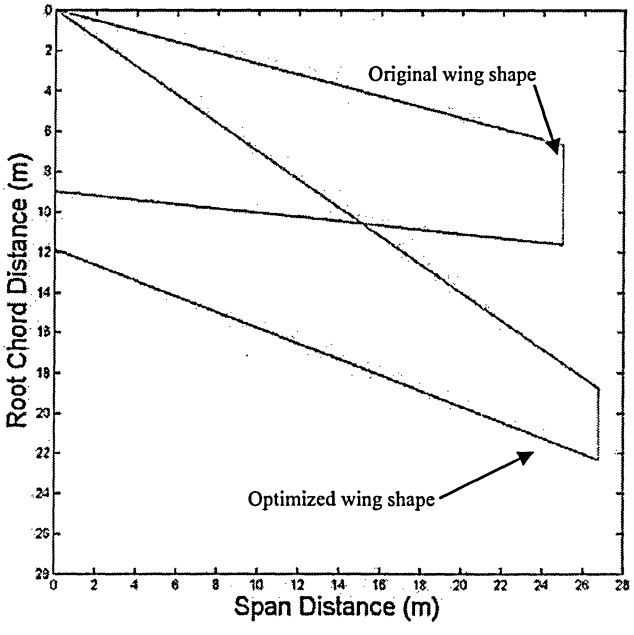


Figure 5-5: Original and optimized rigid wing shape

The overall optimization started at the initial points given in Table 5-7 and resulted in an initial range of 3,228 km. The maximized range was calculated to be 10,224 km which is an improvement of 3.16 times over the original range. The convergence graph of the range can be seen in Figure 5-6. The smooth increase in the range of the aircraft is attributed to consistent derivative calculations used in the optimization. The overall range was maximized in 25 iterations of the optimization process. The final 11 iterations of the optimization was done to enforce both the lift equals weight constraint and the stress constraint.

Table 5-7: Rigid wing design variable bounds, initial and optimum points

	Lower Bound	Upper Bound	Initial	Optimal
Aerodynamic Design Variables				
Sweep (deg)	0	35	15	35
Taper	0.3	1	0.55	0.3
Root Chord (m)	6	13	9	11.87
Span (m)	20	35	25	26.78
t/c	0.1	0.16	0.12	0.1
Structural Design Variables				
Root Upper Skin Thickness (mm)	6	30	19	13.36
Root Lower Skin Thickness (mm)	6	30	14	10.98
Root Spar Thickness (mm)	6	40	18	15.34
Root Rib Thickness (mm)	6	40	15	6
Stringer Height (mm)	5	20	10	18.18
Stringer Width (mm)	5	20	10	13.18
Linear Structural Taper	0.1	0.9	0.3	0.1

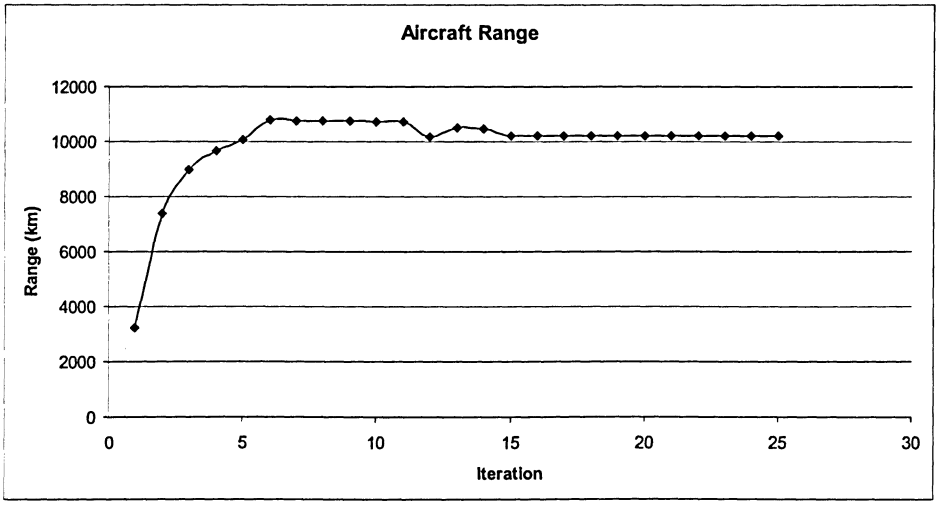


Figure 5-6: Rigid wing convergence of the optimization objective function

The aerodynamic variables, sweep, taper, and t/c were minimized to their lower bounds to give the most aerodynamically efficient configuration. The aerodynamic variables, root chord and semi-span, were primarily used to accommodate the lift equals weight constraint. The convergence of the lift equals weight constraint can be seen in Figure 5-8. The aspect ratio of the wing was not maximized because of the structural limits of the wing. Figure 5-7 provides a comparison of the pressure distributions of the original wing shape versus the optimized wing shape. Note that in Figure 5-7 the

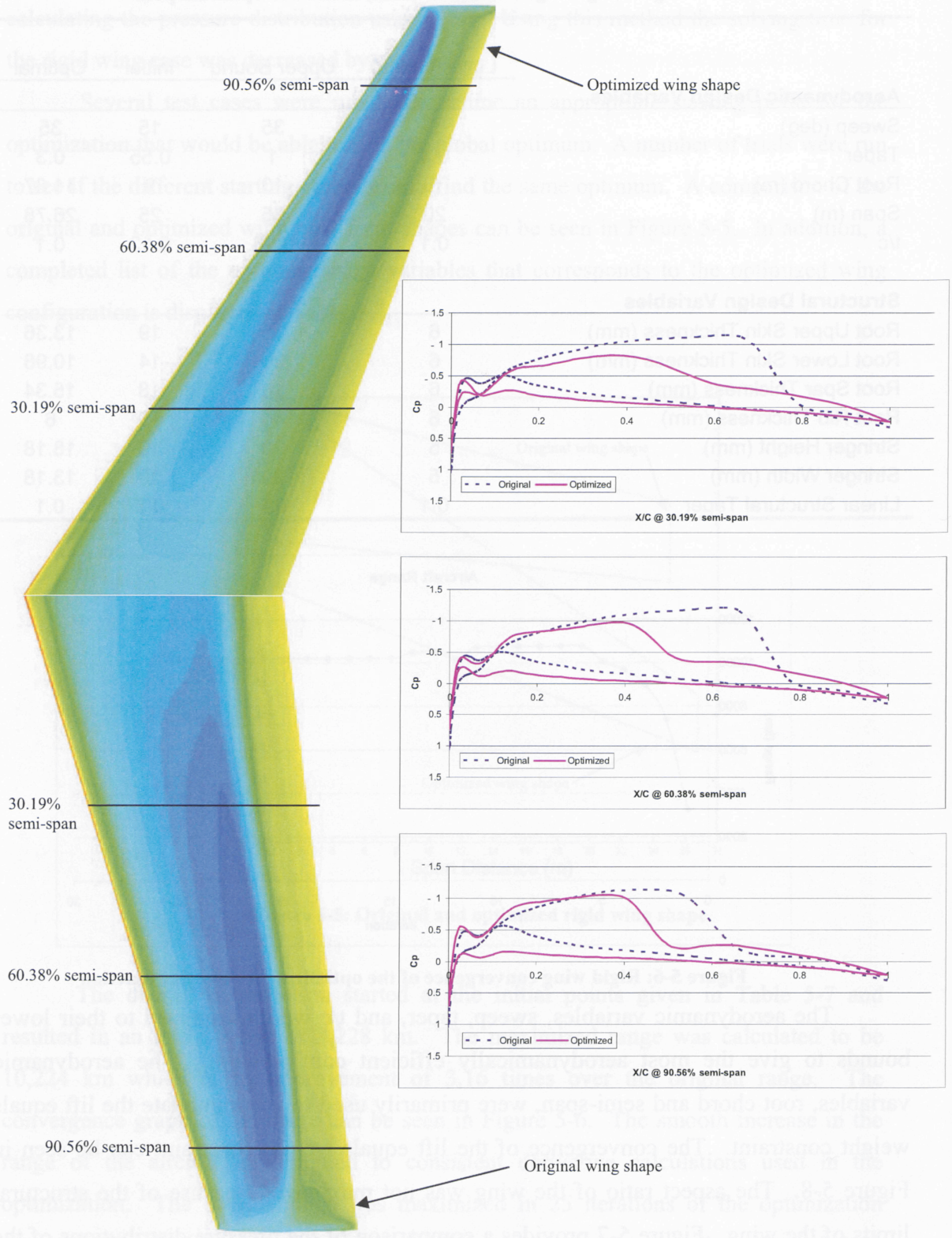


Figure 5-7: Pressure distribution of the original and optimized wing shapes

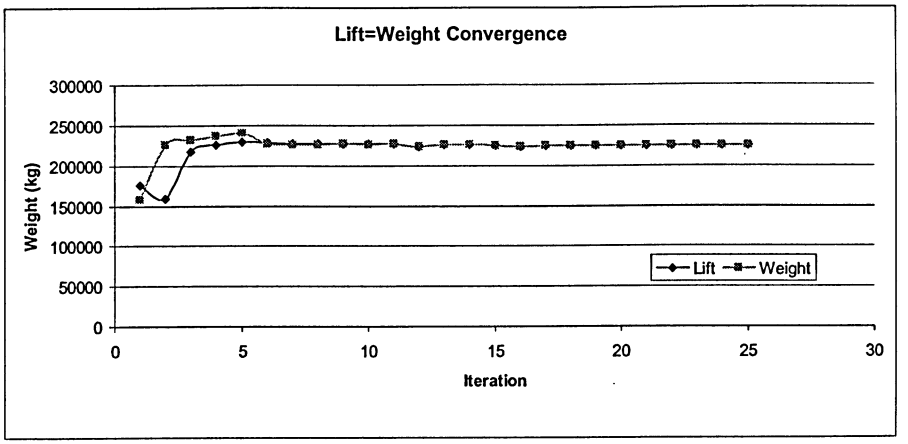


Figure 5-8: Rigid wing convergence of lift equals weight constraint

optimized wing has been scaled to make the comparison between wings easier. Selected stations along the wing span are used to compare the pressure distributions. At each station on the wing the strength of the shock wave has been decreased when compared to the original wing shape. This is most visible when considering the inner most pressure distribution of the wing. The increase in the L/D ratio seen in Figure 5-9 is primarily due to the decrease in the strength of the shock wave on the top of the wing.

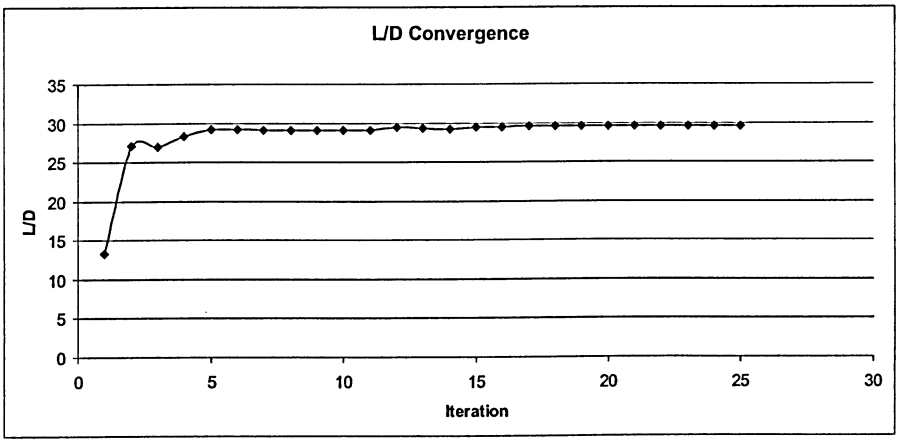


Figure 5-9: Rigid wing convergence of lift-to-drag ratio

The main purpose of the structural element optimization was to find the best configuration that would keep the stress below the stress constraint while minimizing the mass. The structural design variables that had active constraints were the linear structural taper and the rib thickness. The rib was minimized to its lower bound because in this

model the rib is primarily just used to prevent buckling of the skins and it not a major stress bearing element. The linear structural taper was also minimized to its lower bound primarily because it decreased the mass of the wing. It was also minimized because the loads near the tip of the wing are small and thus the structural elements can correspondingly be small.

The structural design variables, upper skin thickness, lower skin thickness, spar thickness, stringer height, and stringer width were optimized to non-active values. The optimized values showed a similar trend when compared to results from a similar study [36]. The upper skin was optimized to be higher than the lower skin and the spar thickness was higher than both. The upper skin was expected to be thicker than the lower skin primarily because the distance from the wing box neutral axis was higher for the upper skin, which results in a higher stress. The spar thickness was expected to be thicker than both the skin thicknesses because it has a large effect on the wing inertia with a small increase in mass. The stringer height and width found optimal points that tended to maximize the stringer inertia to help decrease the root stress.

The effect of changing the structural variables can be seen directly when considering the stress convergence graph (Figure 5-10) along with the wing weight convergence graph (Figure 5-11). Close examination of the two graphs reveals that careful structural optimization can drastically decrease the wing stress with very little mass impact. This is evident when considering the data points between iteration 10 and iteration 20 for all of the parameter convergence graphs. The change between iterations 10 and 20 for the L/D convergence, the lift equals weight convergence, and the wing weight convergence is small. However, there are still large perturbations with the stress constraint, and the stress constraint is still in violation at iteration 10. By iteration 20 the stress constraint is no longer in violation while the wing structural mass has not increased much. This means that small alterations in the structural variables decreased the wing stress by approximately 25 MPa and does not have a large effect on the overall wing weight. The slight bump in the stress convergence occurring at iteration 13 is reflected in the overall range convergence graph (Figure 5-6). It shows that the range was slightly increased due to thinner structural elements but the stress constraint was in violation. The mass of the optimal rigid wing converged to 9,363 kg.

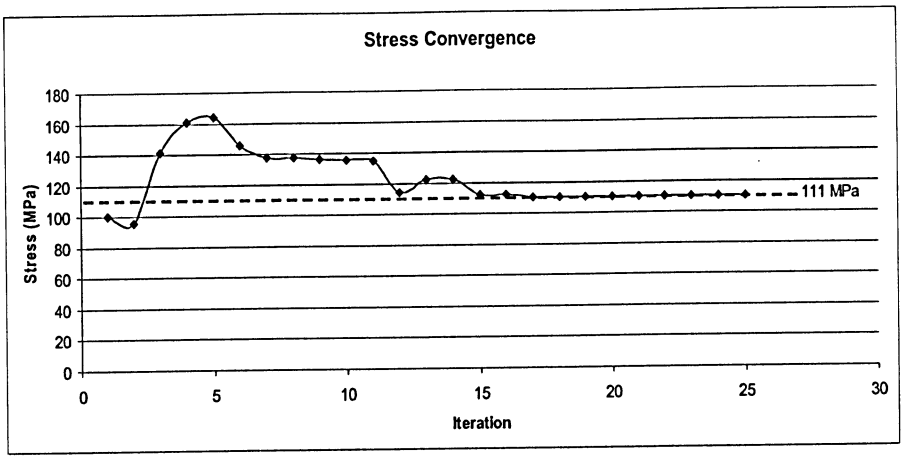


Figure 5-10: Rigid wing convergence of stress constraint

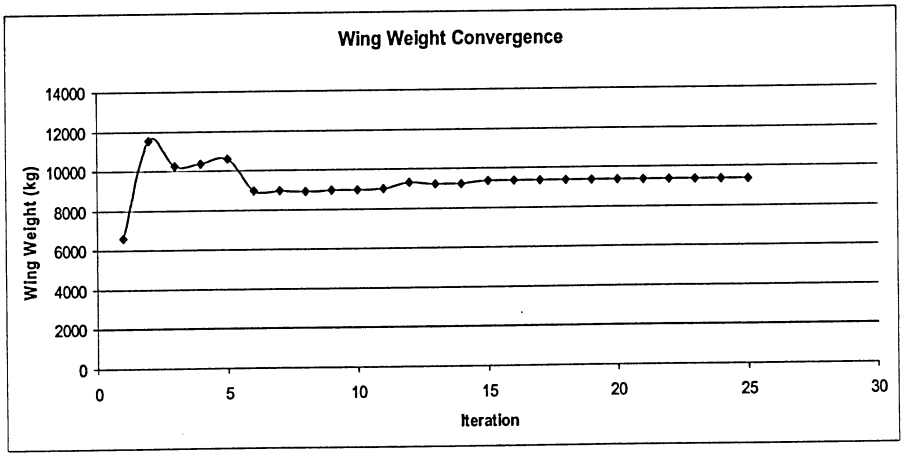


Figure 5-11: Rigid wing convergence of total wing weight

5.5.2 Flexible Wing Results

The optimization of the flexible wing was completed using the same aerodynamic design variables and structural design variables that were used in the rigid wing optimization. The difference between the flexible wing optimization and rigid wing optimization is that for the flexible wing optimization the structural design variables affect the aerodynamics of the wing. This is because changes in the wing box structure can cause changes in the wing tip deflection and wing tip rotation. For this reason the pressure data base that was used in the rigid wing case could not be used. This means that for every single derivative calculation during the optimization a complete static

aeroelastic solution needs to be found and there are no short cuts available. The only way to decrease the solving speed was to implement an under relaxation factor as well as use non-converged CFD solution during the early static aeroelastic iterations. For all the static aeroelastic solutions found during the flexible optimization a converged wing shape was found where the maximum point of deflection occurred at the trailing edge.

To find appropriate starting points for the flexible wing optimization a single optimization was done to approximate the optimal design point. The trial optimization was cut short when approximate optimums were found and would yield appropriate starting points. A comparison of the rigid original, rigid optimized and flexible optimized wing shapes can be seen in Figure 5-12. The corresponding optimal design points for the optimized flexible wing shape are given in Table 5-8.

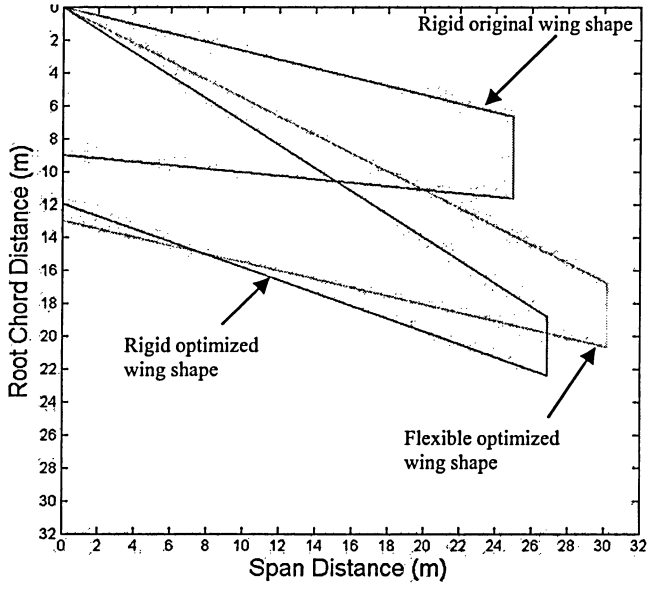


Figure 5-12: Original, optimized rigid, and optimized flexible wing shapes

The flexible optimized wing shape that is shown in Figure 5-12 resulted in a tip deflection of 1.47 m and a tip pitch of -1.34°. The optimization started at the initial points given in Table 5-8 which resulted in an initial aircraft range of 8,994 km. The optimized range that was calculated by the optimizer was 11,292 km which is an improvement of 1.25 versus the initial range. The convergence of the aircraft range can be seen in Figure 5-13. The overall smooth trend that is seen in the range convergence is

Table 5-8: Flexible wing design variable bounds, initial and optimum points

	Lower Bound	Upper Bound	Initial	Optimal
Aerodynamic Design Variables				
Sweep (deg)	0	35	30	29.07
Taper	0.3	1	0.3	0.3
Root Chord (m)	11	13	12.9	12.97
Span (m)	26	34	28.34	30.16
t/c	0.1	0.12	0.109	0.1
Structural Design Variables				
Root Upper Skin Thickness (mm)	5	30	13.73	12.99
Root Lower Skin Thickness (mm)	5	30	26.17	11.02
Root Spar Thickness (mm)	5	30	30	30
Root Rib Thickness (mm)	5	9	8	8.43
Stringer Height (mm)	5	20	17.26	15.53
Stringer Width (mm)	5	20	5	8.65
Linear Structural Taper	0.1	0.12	0.1	0.1075

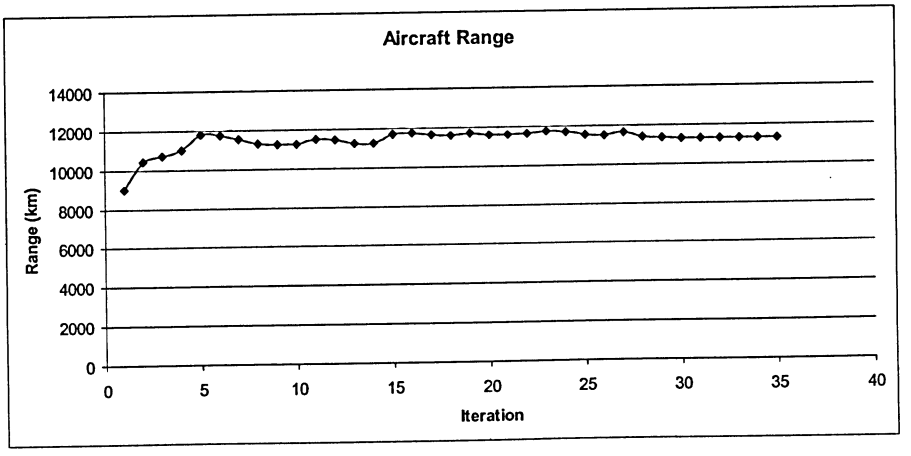


Figure 5-13: Flexible wing convergence of the optimization objective function

once again due to consistent derivative calculations. The overall range was maximized in 35 iterations of the optimizer. The increase in the amount of iterations was due to the complex nature of the flexible optimization. For the rigid optimization only the aerodynamic design variables had an effect on the aerodynamics; however for the flexible wing the changes in the structures affect the aerodynamics as well. This resulted in a more complicated mathematical model with more local maximums. Another difference between the rigid and flexible wing optimizations is that the flexible wing resulted in a higher range when compared to the rigid case. This is not what was initially expected because the flexible wing was thought to have worse aerodynamic characteristics than the

rigid wing and thus result in a decrease in the overall range. The change in the aerodynamic properties of the flexible wing did result in worse aerodynamic properties when compared against the rigid case. This can be seen when comparing the pressure distributions on the rigid and flexible wings as shown in Figure 5-15. The total amount of lift that is created by the flexible wing is decreased when compared against the rigid wing. This is due to the negative pitch of the wing tip that decreases the local angle of attack for the outboard wing sections. This is most evident when considering the pressure distribution at the 90.56% semi-span station. The increase in the overall range was found to be due to the increase in the fuel capacity of the wing as a result of the change in the stresses for the flexible wing. Details concerning this are discussed later.

The aerodynamic design variables found optimal points that are different than the rigid wing case. The sweep of the wing could no longer be maximized because of the need to enforce the lift equals weight constraint. Increasing the wing sweep will help improve the L/D characteristics of the wing but it will also decrease the total lift generated by the wing. Consequently, in order to increase the lift of the wing so that the lift equals weight is satisfied the wing sweep could not be maximized. The remaining design variables, namely span and root chord, were set to also help match the lift equals weight constraint. The overall lift equals weight constraint convergence can be seen in Figure 5-14. The L/D characteristics of the wing were found early on in the optimization

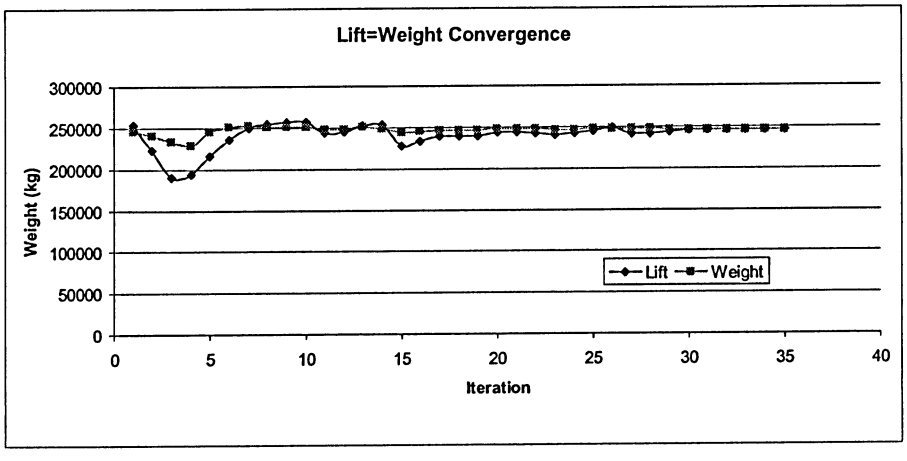


Figure 5-14: Flexible wing convergence of lift equals weight constraint

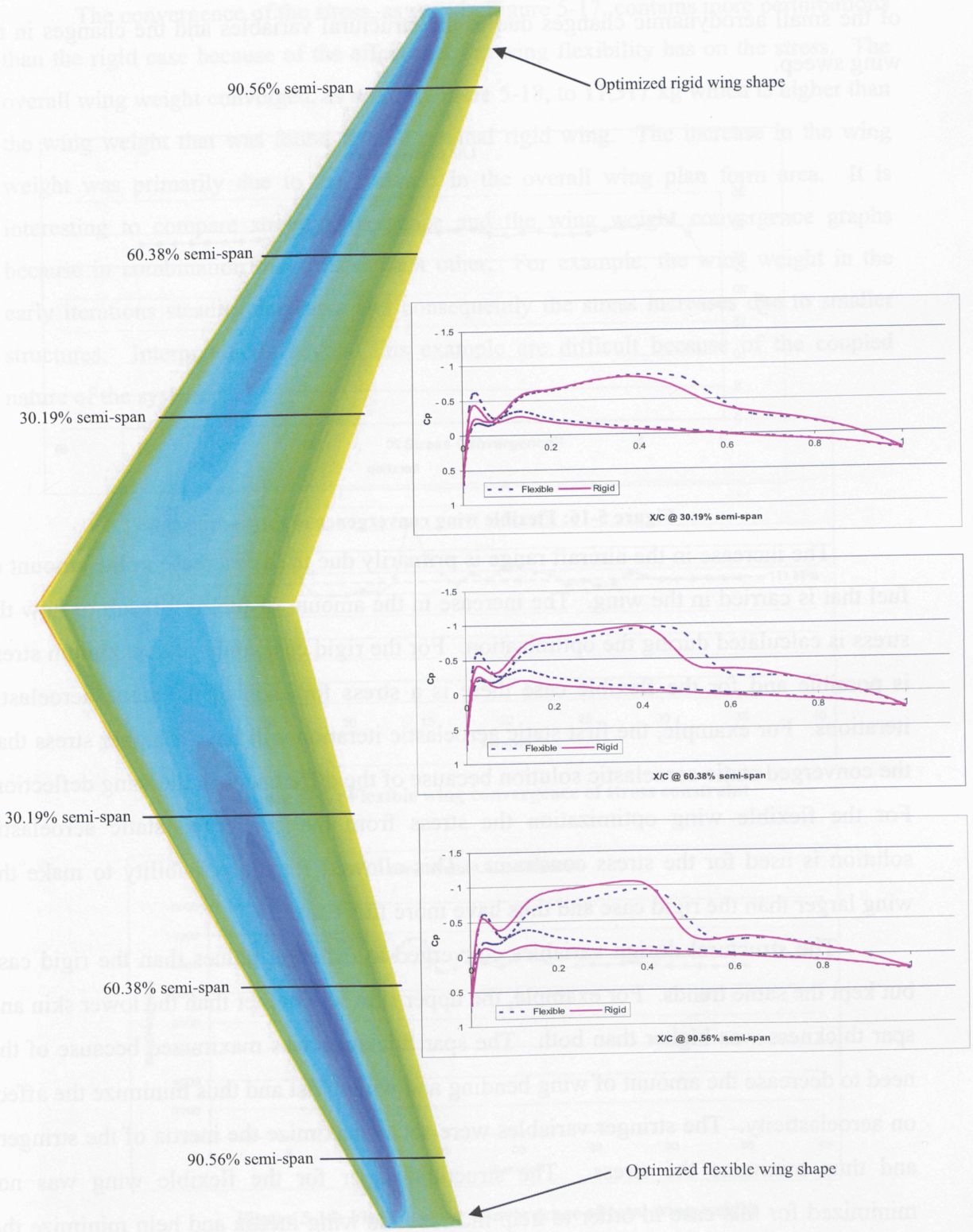


Figure 5-15: Pressure distribution of the optimized rigid and optimized flexible wing shapes

process as seen in Figure 5-16. The small perturbations in the convergence are because of the small aerodynamic changes due to the structural variables and the changes in the wing sweep.

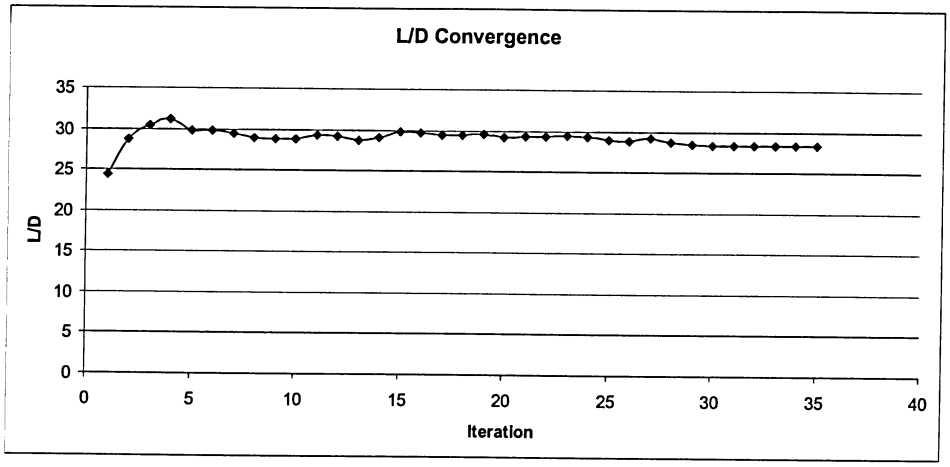


Figure 5-16: Flexible wing convergence of lift-to-drag ratio

The increase in the aircraft range is primarily due to the increase in the amount of fuel that is carried in the wing. The increase in the amount of fuel is a result of how the stress is calculated during the optimization. For the rigid case, only one maximum stress is possible and for the flexible case there is a stress for each of the static aeroelastic iterations. For example, the first static aeroelastic iteration will have a higher stress than the converged static aeroelastic solution because of the differences in the wing deflection. For the flexible wing optimization the stress from the converged static aeroelastic solution is used for the stress constraint. This allowed for the possibility to make the wing larger than the rigid case and thus have more fuel capacity.

The structural design variables converged to different values than the rigid case but kept the same trends. For example, the upper skin was higher than the lower skin and spar thickness was higher than both. The spar thickness was maximized because of the need to decrease the amount of wing bending and wing twist and thus minimize the affect on aeroelasticity. The stringer variables were set to maximize the inertia of the stringers and thus minimize the stress. The structural taper for the flexible wing was not minimized for this case in order to help increase the wing inertia and help minimize the aeroelastic effects.

The convergence of the stress, as seen in Figure 5-17, contains more perturbations than the rigid case because of the effect that the wing flexibility has on the stress. The overall wing weight converged, as seen in Figure 5-18, to 11,311 kg which is higher than the wing weight that was found for the optimal rigid wing. The increase in the wing weight was primarily due to the increase in the overall wing plan form area. It is interesting to compare stress convergence and the wing weight convergence graphs because in combination they reflect each other. For example, the wing weight in the early iterations steadily decreases and consequently the stress increases due to smaller structures. Interpretations beyond this example are difficult because of the coupled nature of the system.

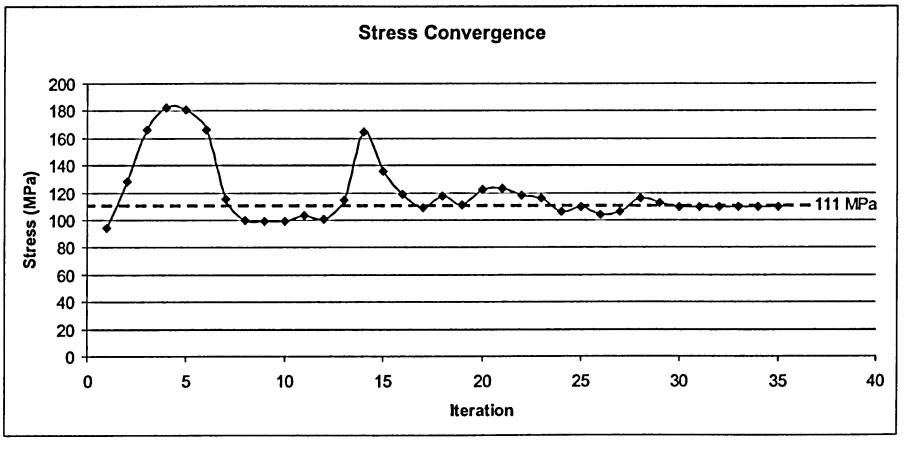


Figure 5-17: Flexible wing convergence of stress constraint

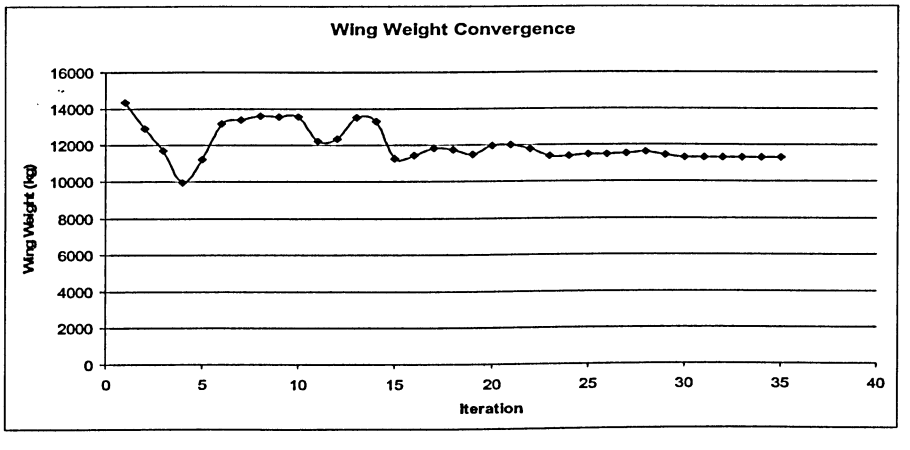


Figure 5-18: Flexible wing convergence of total wing weight

Chapter 6

Conclusion and Future Work

The purpose of this thesis was to determine the differences between rigid and flexible optimized wing shapes while considering high-fidelity aerodynamics and high-fidelity structures simultaneously. For the high-fidelity aerodynamics analysis, FLUENT was chosen and used to obtain pressure distributions along the wing. For the high-fidelity structural analysis, a finite element model was created in ANSYS to obtain the structural responses of the wing.

In order to take both the aerodynamic and structural disciplines into consideration MDO was implemented. After an extensive review of the five main MDO architectures it was found that the MDF method was the simplest and could be used for the optimization of the high-fidelity wing. In order to use the MDF architecture a method of coupling the high-fidelity analysis packages together was developed. Using MATLAB as a wrapper a computational static aeroelastic solver was developed that can find the static wing shape of any wing configuration. It was found that the wall clock time to solve the static aeroelastic problem could be sped up using the following techniques. The implementation of an under relaxation factor could be used to help remove the oscillations in the solution and also speed up the convergence. A second method of speeding up the solution was to use non-converged CFD solutions to get approximations of the structural response for early static aeroelastic iterations. A fully converged CFD solution only needs to be solved on the last static aeroelastic iteration.

The computational static aeroelastic solver was combined with the MDF-MDO architecture and SQP was used as an optimizer. Optimizer was given the task of maximizing the range of the aircraft based on the Breguet range equation by changing the wing shape. Optimizations were performed using several aerodynamic and structural design variables on rigid and flexible wings. The design variables were chosen such that as much freedom was given to the optimizer to choose the optimal wing. The optimizer was successful in finding optimal wing designs for both the rigid and flexible cases. The optimal range for the rigid and flexible wings was found to be 10,224 km and 11,292 km respectively. Overall, it was found that for the optimization to be successful using gradient based optimizers, the most important aspect of the entire process is the accurate calculation of the sensitivities. If the gradients of the design space cannot be calculated properly then the entire optimization will not be successful.

There are many things that can be done to improve the accuracy of the work that is done for this thesis, as this is just a first step in a complex process. The first thing that can be done is to increase the CFD grid density, to allow for a more accurate pressure distribution to be calculated. It would also be of interest to improve the methods of mapping the CFD grid to the FEM mesh, and improve the method of calculating the FEM nodal forces. To help fix both of these problems a NURBS surface should be used for the transfer and application of the forces. The NURBS surface would allow the forces to be applied to any node while maintaining load conservation. It would also be beneficial to create a more robust automatic CFD grid maker so a grid of maximum quality could be created for any wing configuration. In addition, a dynamic mesher could also be implemented to automatically change the shape of the aeroelastic wing, while maintaining grid quality.

The next step in high-fidelity optimization is to expand the number of design variables that are optimized. Increasing the number of structural design variables to include sizing parameters for each of the structural elements would be interesting because a more detailed structure could be designed. Additionally, the number of aerodynamic design variables should be increased. Adding a design variable for wing twist would allow for a more realistic aeroelastic design. It would also be interesting to create several design variables for each airfoil section along the wing. This way each airfoil section on

the wing will be different and a true aerodynamic optimum could be found. A separate study should be completed where changes in the optimization constraints are made. This would help determine the effect that the constraints have on the optimized wing configuration. For example, it would be good to compare wing configurations where the lift equals weight constraint is not included. Finally, it would be beneficial to attempt to use these methodologies for wing optimization for off design cases. For example, it would be good to know how the wing design would change when it is designed for an entire mission.

The future of reliable high-fidelity optimization depends on decreasing the amount of numerical noise that exists with CFD calculations. A method to decrease the amount of noise is to increase the CFD grid density. This will decrease the amount of low amplitude high frequency noise that the CFD analysis has. Another approach to decrease the amount of noise is to introduce the use of response surfaces to effectively filter the noise from the CFD.

In the future it would be interesting to take a step back and resort to intermediate-fidelity models and apply the IDF and CO MDO architectures. Researchers in this field have yet to successfully apply the IDF method or the CO method for intermediate/high-fidelity wing optimization. This is an extremely challenging topic because of the task of matching piecewise lift, drag, and moment values along the wing. Using intermediate-fidelity will allow the optimization to converge quicker versus high-fidelity and thus allow optimization changes to be made more rapidly. Unfortunately, the decoupled natures of both the IDF and CO methods cannot be taken advantage of at this time for applications in high-fidelity optimization until the time required to calculate the aerodynamic sensitivities has drastically decreased.

The key to an efficient gradient based optimization is accurate and timely derivative calculations. It would be beneficial to introduce different fidelity models into the optimization procedure and test the speed and accuracy of the derivative calculations using the lower-fidelity models. Another alternative is to take advantage of using the global sensitivity equations or the coupled-adjoint method for efficiently calculating sensitivities.

In addition, a piece of software should be created that can be used to graphically view the optimization procedure. It would be good if the software had the ability to optimize a wing based on different MDO architectures and give the user plenty of choice in choosing design variables.

Lastly, the future of flight will depend on highly efficient aircraft that will be able to change shape while in flight. This type of optimization could be combined with morphing wing technology and smart structures to perform wing optimization in real time. Adapting fast optimization schemes to these technologies has the potential to drastically improve aircraft performance.

References

[1] Renaud, G., Chen, S., Shi, G., Zhang, S., Yang, S., 'An Application of Multidisciplinary Feasible (MDF) Formulation for Aircraft Wing Design', National Research Council, Institute for Aerospace Research, Ottawa, Ontario, Canada, 2002.

[2] Raymer, D. P., Aircraft Design: A Conceptual Approach, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1999.

[3] Pamadi, B. N., Performance, Stability, Dynamics, and Control of Airplanes, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1998.

[4] MATLAB V7.0R14 User Guide, The MathWorks Inc., 3 Apple Hill Drive, Natick, MA, 2005.

[5] GAMBIT V2.1.2 User Manual, FLUENT Inc., 10 Cavendish Court, Lebanon, NH 03766, USA.

[6] FLUENT V6.1 User Manual, FLUENT Inc., 10 Cavendish Court, Lebanon, NH 03766, USA.

[7] ANSYS V8.0 User Manual, ANSYS Inc., Southpointe, 275 Technology Drive, Canonsburg, PA 15317, USA.

[8] Sobieszczanki-Sobieski, J., Haftka, R.T., 'Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments', AIAA, 1997.

[9] AIAA Technical Committee on Multidisplinary Design Optimization. 'White Paper on Current State of the Art', AIAA, 1991.

[10] Alexandrov, N, M., Lewis, R. M., 'Comparative properties of collaborative optimization and other approaches to MDO' Conference on Engineering Design Optimization, July 8-9, 1999.

[11] Kodiyalam, S., Sobieszczanki-Sobieski, J., 'Multidisciplinary Design Optimization – some formal methods, framework requirements, and application to vehicle design'. Int. J. Vehicle Design (Special Issue), pp. 3-22.

[12] Cramer, E. J., Dennis Jr, J. E., Frank, P. D., Lewis, R. M., Shubin, G. R., 'Problem Formulation for Multidisciplinary Optimization'. SIAM Journal on Optimization, No. 4, pp. 754-776, November 1994.

[13] Hulme, K. F., Bloebaum, C. L., 'A Comparison of Solution Strategies for Simulation-based Multidisciplinary Design Optimization', American Institute of Aeronautics and Astronautics, AIAA Paper 98-4977, 1998.

[14] Alexandrov, N. M., Kodiyalam, S., 'Initial Results of an MDO Method Evaluation Study'. American Institute of Aeronautics and Astronautics, AIAA Paper 98-4884, 1998.

[15] Sellar, R. S., Batill, S. M., Resaud, J. E., 'Response Surface Based, Concurrent Subspace Optimization for Multidisciplinary System Design, American Institute of Aeronautics and Astronautics, AIAA Paper 96-0714, 1996.

[16] Braun, R., Gage, P., Kroo, I., Sobieski, I., 'Implementation and performance issues in Collaborative Optimization', NASA-AIAA-96-4017, 1996.

[17] McAllister, C. D., Simpson, T. W., Lewis, K., Messac, A., 'Robust multiobjective optimization through Collaborative Optimization and linear physical programming', Multidisciplinary Analysis and Optimization Conference, AIAA 2004-4549, 2004.

[18] Kroo, I., Altus, S., Bruan, R., Gage, P., Sobieski, I., 'Multidisciplinary Optimization Methods for Aircraft Preliminary Design'. AIAA 94-4325, 1994.

[19] Alexandrov, N. M., Lewis, R. M., 'Analytical and computational aspects of collaborative optimization'. NASA/TM-2000-210104, 2000.

[20] Sobieszczanski-Sobieski, J., Jeremy S. Agte and Robert R. Sandusky, Jr.,'Bi-Level Integrated System Synthesis (BLISS)', NASA/TM-1998-208715, August 1998.

[21] Sobieszczanski-Sobieski, J., Emiley, M. S., Agte, J. S., Sandusky, R. R., 'Advancement of Bi-Level Integrated System Synthesis (BLISS)', AIAA 2000-0421, 2000.

[22] Sobieszczanski-Sobieski, J., Kodiyalam, S., 'BLISS/S: a new method for two-level structural optimization', Structural Multidisciplinary Optimization. Vol. 21, pp. 1-13. 2001.

[23] Perez, R. E., Liu, H. H. T., Behdinan, K., 'Evaluation of Multidisciplinary Optimization Approaches for Aircraft Conceptual Design', Proceedings Tenth AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA Paper 2004-4537, Albany, New York, September 2004.

[24] Sobieszczanski-Sobieski, J. 'Sensitivity of Complex, Internally Coupled Systems', AIAA Journal, January, Vol. 28, no. 1. pp. 153-160. 1994.

[25] Kodiyalam, S., Yuan, C., 'Evaluation of Methods for Multidisciplinary Design Optimization (MDO), Part II', NASA/CR-2000-210313, November 2000.

[26] Hölinger, H. G., Krammer, J., Stettner, M., 'MDO Technology Needs in Aeroelastic Structural Design', AIAA paper 98-4731, 1998.

[27] Lamberti, L., Pappalettere, C., 'Design optimization of large-scale structures with sequential linear programming', Proceedings of the I MECH E Part C Journal of Mechanical Engineering Science. Vol. 216, no. 8, pp. 799-811. August 2004.

[28] Lund, E., Moller, H., Jakobsen, L, A., 'Shape optimization of Fluid-Structure Interaction Problems using Two-Equation Turbulence Models', Proceedings 43rd AIAA/ASME.ASCE/AHS/ASC Symposium on Sturctures, Structural Dynamics, and Materials. AIAA Paper 2002-1478 Denver, April 22-25, 2002.

[29] Sarker, R. A., Gunn, E, A., 'A Simple SLP algorithm for solving a class of nonlinear programs'. European Journal of Operational Research. Vol. 101, pp. 140-154. 1997.

[30] Vanderplaats, G. N., Numerical Optimization Techniques for Engineering Design. McGraw-Hill, New York, 1984.

[31] Venkataraman, P., Applied Optimization with MATLAB Programming. John Wiley & Sons, Inc., New York, 2002.

[32] Zhu, Z., 'An efficient sequential quadratic programming algorithm for nonlinear programming'. Journal of Computational and Applied Mathematics. Vol. 175, pp. 447-464. 2005.

[33] Shanno, D. F., Phua, K. H., 'Numerical Experience with Sequential Quadratic Programming Algorithms for Equality Constrained Nonlinear Programming'. ACM Transactions on Mathematical Software. Vol. 15, No. 1, pp. 49-63. March 1989.

[34] Nocedal, J., Wright, S. J., Numerical Optimization. Springer-Verlag, New York, 1999.

[35] Gumbert, C, R., Hou, G, J, W., Newman, P, A., 'Simultaneous Aerodynamic Analysis and Design Optimization (SAADO) for a 3-D Flexible Wing', AIAA Paper 2001-1107, January 2001.

[36] Chen, S., Zhang, S., Renaud, G., Shi, G., Yang, X., 'A Preliminary Study of Wing Aerodynamic, Structure, and Aeroelastic Design and Optimization', 9th AIAA/ISSMO Symposium of Multidisciplinary Analysis and Optimization, AIAA-2002-5656, Atlanta, Georgia, USA. Sept. 4-6, 2002.

[37] Giesing, J. P., Barthelemy, J. M., 'A Summary of Industry MDO Applications and Needs', AIAA Paper 98-4737, September 1998.

[38] Martins, J. R. R. A., 'A Coupled-Adjoint Method For High-Fidelity Aero-Structural Optimization' Ph.D. thesis, Stanford University, Standford, California, October 2002.

[39] Perez, R. E., Liu, H. H. T., Behdinan, K., 'Flight Dynamics and Control Multidisciplinary Integration in Aircraft Conceptual Design Optimization', Proceedings Tenth AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA Paper 2004-4435, Albany, New York, September 2004.

[40] Perez, R. E., Liu, H. H. T., Behdinan, K., 'Early Aircraft and Control Design Integration through Multidisciplinary Optimization and Surrogate Models', Proceedings AIAA Guidance, Navigation, and Control Conference and Exhibit, AIAA Paper 2004-5356, Providence, Rhode Island, August 2004.

[41] Wakayama, S., Kroo, I., 'The Challenge and Promise of Blended-Wing-Body Optimization', AIAA Paper 98-4736, September 1998.

[42] Gumbert, C, R., Hou, G, J, W., Newman, P, A., 'Simultaneous Aerodynamic Analysis and Design Optimization (SAADO) for a 3-D Rigid Wing', in Proceedings of 14th AIAA Computational Fluid Dynamics Conference, Norfolk, pp. 402-418 / AIAA paper 99-3296, June 1999.

[43] Gumbert, C, R., Hou, G, J, W., Newman, P, A., 'High-Fidelity Computational Optimization for 3-D Flexible Wings: Part 1 – Simultaneous Aero-Structural Design Optimization (SASDO)', Optimization and Engineering, 6, 117-138, 2005.

[44] Gumbert, C, R., Hou, G, J, W., Newman, P, A., 'High-Fidelity Computational Optimization for 3-D Flexible Wings: Part II—Effect of Random Geometric Uncertainty on Design', Optimization and Engineering, 6, 139-156, 2005.

[45] Reuther, J., Alonso, J. J., Martins, J. R. R. A., Smith, S. C., ' A Coupled Aero-Structural Optimization Method for Complete Aircraft Configurations', AIAA Paper 99-0187, 1999.

[46] Martins, J. R. R. A., Alonso, J. J., Reuther, J., 'Aero-Structural Wing Design Optimization Using High-Fidelity Sensitivity Analysis', CEAS Conference on Multidiciplinary Analysis and Optimization. Cologne, Germany. June 2001.

[47] Alonso, J. J., LeGresley, P., Van der Weide, E., Martins, J. R. R. A., Reuther, J. J., 'pyMDO: A Framework for High-Fidelity Multi-Disciplinary Optimization', AIAA Paper 2004-4480, September 2004.

[48] Maute, K., Allen, M., 'Conceptual Design of Aeroelastic Structures by Topology Optimization', Structural Multidisciplinary Optimization', 27, 27-42, 2004.

[49] Carrier, C., 'Multi-Disciplinary Optimisation of a Supersonic Transport Aircraft Wing Planform', European Congress on Computational Methods in Applied Science and Engineering, July 2004.

[50] Anderson, J. D., 'Fundamentals of Aerodynamics 3rd Edition', McGraw-Hill, New York, 2001.

[51] Bhatia, M., Ajmera, H. C., Abhankar, S. N., Mujumdar, P. M., Sudhakar, K., 'WingOpt – An MDO Tool for Concurrent Aerodynamic Shape and Structural Sizing Optimization of Flexible Aircraft Wings', Indian Institute of Technology Bombay, Technical Report.

[52] Alexandrov, N. M., Nielsen, E. J., Lewis, R. M., Anderson, W. K., 'First-Order Model Management with Variable-Fidelity Physics Applied to Mutli-Element Airfoil Optimization'. AIAA Paper, 2000-4886. 2000.

[53] Le Moigne, A., Qin, N., 'Variable-Fidelity Aerodynamic Optimization for Turbulent Flows Using a Discrete Adjoint Formulation', AIAA Journal, Vol. 42, No. 7, pp. 1281-1292. December 2003.

[54] Schmit, Jr., L. A., Farshi, B., 'Some approximation concepts for structural synthesis'. AIAA Journal, Vol. 12, No. 5, pp. 692-699. 1974.

[55] Alexandrov, N. M., Lewis, R. M., Grumbert, C. R., Green, L. L., Newman, P. A., 'Optimization with Variable-Fidelity Models Applied to Wing Design', AIAA Paper 2000-0841, Jan. 2000.

[56] Samareh, J, A., Bhatia, K. G., 'A Unified Approach to Modeling Multidisciplinary Interactions', AIAA Paper 2000-4704, 2000.

[57] Ahrem, R., Beckert, A., and Wendland, H., 'A New Multivariate Interpolation Method for Large-Scale Spatial Coupling Problems in Aeroelasticity', 2001.

[58] Beckert, A., 'Coupling fluid (CFD) and Structural (FE) Model using Finite Interpolation elements', Aerospace, Science, and Technology (AST), No. 5082, No. 4, pp. 13-22. 2000.

[59] Bhardwaj, M, K., 'A CFD/CSD Interaction Methodology for Aircraft Wings', Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksberg, Virginia, October 1997.

[60] Airfoil Geometry, http://www.desktopaero.com/appliedaero/airfoils1/airfoilgeometry.html, August 2005.

[61] Samareh, J, A., 'Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization', AIAA Journal, Vol. 39, No. 5, pp. 877-884. May 2001.

[62] Samareh, J, A., 'Novel Multidisciplinary Shape Parameterization Approach',Journal of Aircraft, Vol. 38, No. 6, pp. 1015-1024. November-December 2001.

[63] Walsh, J. L., Townsend, J. C., Salas, A. O., Samareh, J. A., Mukhopadhyay, V., Baethelemy, J.-F., 'Multidisciplinary High-Fidelity Analysis and Optimization of Aerospace Vehicles, Part 1: Formulation', AIAA Paper 2000-0418, 2000.

[64] Walsh, J. L., Weston, R. P., Samareh, J. A., Mason, B. H., Green, L. L., Biedron, R. T., 'Multidisciplinary High-Fidelity Analysis and Optimization of Aerospace Vehicles, Part 2: Preliminary Results', AIAA Paper 2000-0419, 2000.

[65] Hosder, S., Watson, L. T., Grossman, B., Mason, W., Haftka, R. T., Cox, S. E., 'Polynomial Response Surface Approximations for the Multidisciplinary Design Optimization of a High Speed Civil Transport', Optimization and Engineering, Vol. 2, 2001, pp. 431-452, 2001.

[66] Niu, M. C., 'Airframe Structural Desing: Practical Design Information and Data on Aircraft Structures', Conmilit Press Ltd., Hong Kong, 1999.

[67] Anhalt, C., Monner, H. P., Breitback, E., 'Interdisciplinary Wing Design – Structural Aspects', SAE International, 03WAC-29, 2003.

[68] Girodroux-Lavigne, P., Grisval, J. P., Guillemot, S., Henshaw, M., Karlsson, A., Selmin, V., Smith, J., Teupootahiti, E., Winzell, B., 'Comparison of Static and Dynamic Fluid-Structure Interataction Solutions in the Case of a Highly Flexible Modern Transport Aircraft Wing', Aerospace Science and Technology, Vol. 7, pp. 121-133. 2003.

[69] Bennett Associates, 'http://www.bennettmg.co.uk/News/news_Yorkshire_Engineers_Top_Award.html', February 27, 2006.

[70] Bhardwaj, M. K., Kapania, R. K., Reichenbach, E., Guruswamy, G. P., 'Computational Fluid Dynamics/Computational Structural Dynamics Interaction Methodology for Aircraft Wings', AIAA Journal, Vol. 36, No. 12, pp. 1279-2186. December 1998.

[71] Obyashi, S., Guruswamy, G. P., 'Convergence Acceleration of a Navier Stokes Solver for Efficient Static Aeroelastic Computations', AIAA Journal, Vol. 33, No. 6, pp. 1134-1141. June 1995.

[72] Tzong, G., Chen, H. H., Chan, K. C., Wu, T., Cebeci, T., 'A General Method for Calculating Aero-Structure Interaction of Aircraft Configurations', AIAA Paper 96-3982, September 1996.

[73] Chipman, R., Walters, C., MacKenzie, D., 'Numerical Compuation of Aeroelastically Corrected Transonic Loads', AIAA Paper 79-0766, 1979.

[74] Newman III, J.C., Newman, P. A., Taylor III, A. C., Taylor, Hou, G. J.-W., 'Efficient Nonlinear Static Aeroelastic Wing Analysis', Computers and Fluids, Vol. 28, pp. 615-628. 1999.

[75] Cavallo, P, A., 'Coupling Static Aeroelastic Predictions with an Unstructured-Grid Euler/Interacting Boundary Layer Method'. M. Sc. Thesis, The George Washington University, July 1995.

[76] Giunta, A. A., 'Sensitivity Analysis Method for Aeroelastic Aircraft Models', Aircraft Design, Vol. 2, pp. 207-230. 1999.

[77] Abbot, I. H., Von Doenhoff, A. E., 'Theory of wing sections: including a summary of airfoil data', Dover Publications, New York, New York, 1959.

[78] Schmitt, V., Charpin, F., 'Pressure Distributions on the ONERA M6-Wing at Transonic Mach Numbers', Experimental Data Base for Computer Program Assessment, AR-138, AGARD, 1979.

[79] Nielsen, E. J., Anderson, W. K., 'Recent Improvements in Aerodynamics Design Optimization on Unstructured Meshes', AIAA Journal, Vol. 40, No. 6, pp. 1155-1163. 2002.

[80] Lee, B. J., Kim, C. S., Kim, C., Rho, O.-H., Lee, K. D., 'Parallelized Design Optimization for Transonic Wings Using Aerodynamic Sensitivity Analysis', AIAA Paper 2002-0264, January 2002.

[81] Allison, D, O., Cavallo, P. A., 'Static Aeroelastic Predictions for a Transonic Transport Model Using an Unstructured-Grid Flow Solver Coupled with a Structural Plate Technique', NASA/TP-2003-212156, March 2003.

[82] Zeiler, T, A., 'Divergence and Convergence of Iterative Static Aeroelastic Solutions', Journal of Aircraft, Vol. 36, No. 4, pp. 716-718. July-August 1999.

[83] Airliners.net, http://www.airliners.net/info/stats.main?id=106, September 2005.

[84] Staley J. T., and Lege D. J., Advances in Aluminium Alloy Products for Structural Applications in Transportation, Journal de Physique IV, 3, pp. 179-190, November 1993.

[85] Federal Aviation Administration, http://www.faa.gov/, December 2005.

[86] ANSYS 6.1 Documentation, http://www.oulu.fi/atkk/tkpalv/unix/ansys6.1/content /index.html, December 2005.

Appendix A

MDO Numerical Examples

The simple problem that is used as an MDO example is taken from Perez et al. [23]. The analytic problem shown below contains two different disciplines that are described by y_1 and y_2. They are coupled to each other and linked to the overall objective function. The example can be written mathematically as,

$$\begin{aligned} \min \quad & f = x_2^2 + x_3 + y_1 + e^{-y_2} \\ s.t. \quad & \begin{cases} g_1 = \left(\frac{y_1}{3.16}\right) - 1 \geq 0 \\ g_2 = 1 - \left(\frac{y_2}{24}\right) \geq 0 \\ -10 \leq x_1 \leq 10 \\ 0 \leq x_2 \leq 10 \\ 0 \leq x_3 \leq 10 \end{cases} \\ where: & \begin{cases} y_1 = x_1^2 + x_2 + x_3 - 0.2y_2 \\ y_2 = \sqrt{y_1} + x_1 + x_3 \end{cases} \end{aligned} \tag{A-1}$$

This formulation of the problem can be used directly for the MDF method and changed slightly for the IDF and CO methods. The variables defined by x are the global

optimization variables and the variables defined by y are the coupling variables between the two disciplines.

A.1 MDF Numerical Example

The following section will describe a step by step method of the implementation of MDF using the problem that was defined above. To begin the optimization the initial values must be chosen that satisfy a feasible design. For this example the values that were chosen are:

$$x_1 = 1, \qquad x_2 = 5, \qquad x_3 = 2$$

The initial values are used to obtain a multidisciplinary feasible design by solving a system of non-linear equations to obtain feasible coupling variables. This is usually called a system analysis and typically done using Gauss-Seidel procedure (fixed point iteration). For the example above the coupling variables are given by:

$$\left.\begin{array}{l} y_1 = x_1^2 + x_2 + x_3 - 0.2y_2 \\ y_2 = \sqrt{y_1} + x_1 + x_3 \end{array}\right\} \rightarrow \left.\begin{array}{l} y_1 = 1^2 + 5 + 2 - 0.2y_2 \\ y_2 = \sqrt{y_1} + 1 + 2 \end{array}\right\} \rightarrow \begin{array}{l} y_1 = 8 - 0.2y_2 \\ y_2 = \sqrt{y_1} + 3 \end{array}$$

The Gauss-Seidel iterations converge to the following coupling variable values:

$$y_1 = 6.88, \qquad y_2 = 5.62$$

The reason this MDO method is called multidisciplinary feasible is because for each iteration of the optimization there is a feasible design being optimized. The next step is to evaluate the objective function based on the calculated coupling variables (y_1 and y_2) and the selected x variables. For this example a SQP algorithm was used and yielded new x values of:

$$x_1 = 1.9, \qquad x_2 = 0, \qquad x_3 = 1.4$$

The x values are compared to the previous iteration, and if they are found to differ by more than a predetermined ε amount then the MDF procedure has not converged. They can be used on the next iteration as the new feasible design point and the procedure will begin again. If the x values are less than the predetermined ε amount then the MDF

procedure is converged and the optimization is over. For this example the MDF procedure converged after five iterations and resulted in the following optimal points,

$$x_1 = 1.9776, \quad x_2 = 0, \quad x_3 = 0$$
$$y_1 = 3.16, \quad y_2 = 3.7553$$

which match the optimal points given by Perez et al [23].

A.2 IDF Numerical Example

The following section will describe a step by step method of the implementation of IDF using the problem that was defined above. The first step in this MDO method is to add additional design variables for optimization and to add additional constraints for the new design variables. Two new surrogate design variables called x_{y1} and x_{y2} are introduced with two additional equality constraints. The new optimization problem looks similar to the problem seen in equation (A-1) except for the inclusion of the surrogate design variables. The new mathematical formulation is,

$$\begin{aligned} \min \quad & f = x_2^2 + x_3 + x_{y1} + e^{-x_{y2}} \\ s.t. \quad & \begin{cases} g_1 = \left(\frac{y_1}{3.16}\right) - 1 \geq 0 \\ g_2 = 1 - \left(\frac{y_2}{24}\right) \geq 0 \\ x_{y1} - y_1 = 0 \\ x_{y2} - y_2 = 0 \\ -10 \leq x_1 \leq 10 \\ 0 \leq x_2 \leq 10 \\ 0 \leq x_3 \leq 10 \end{cases} \\ where: & \begin{cases} y_1 = x_1^2 + x_2 + x_3 - 0.2x_{y2} \\ y_2 = \sqrt{x_{y1}} + x_1 + x_3 \end{cases} \end{aligned} \tag{A-2}$$

At the beginning of this optimization the values of the design variables are initialized to the following,

$$x_1 = 1, \qquad x_2 = 5, \qquad x_3 = 2 \qquad x_{y1} = 10 \qquad x_{y2} = 4$$

Now the disciplines can be solved directly and a complete system analysis can be avoided.

$$\left.\begin{array}{l} y_1 = x_1^2 + x_2 + x_3 - 0.2x_{y2} \\ y_2 = \sqrt{x_{y1}} + x_1 + x_3 \end{array}\right\} \rightarrow \left.\begin{array}{l} y_1 = 1^2 + 5 + 2 - 0.2 \times 4 \\ y_2 = \sqrt{10} + 1 + 2 \end{array}\right\} \rightarrow \begin{array}{l} y_1 = 7.20 \\ y_2 = 6.16 \end{array}$$

The optimizer now uses the y_1 and y_2 values to optimize for the design variables that were identified above. Each discipline, namely y_1 and y_2, satisfy individual discipline feasibility upon each iteration and they only have complete multidisciplinary feasibility on the last iteration. Overall the optimization process will continue until all of the constraints are satisfied and multidisciplinary feasibility is achieved. The first iteration of the optimization (again SQP) procedure resulted in the design variables having the following values,

$$x_1 = 3.25, \qquad x_2 = 0, \qquad x_3 = 0.54 \qquad x_{y1} = 4.71 \qquad x_{y2} = 6.64$$

The IDF converged after seven iterations but is computationally less intensive because there is no need for system analysis. The optimization converged to the same values at the MDF method.

$$x_1 = 1.9776, \qquad x_2 = 0, \qquad x_3 = 0$$
$$y_1 = 3.16, \qquad y = 3.7553$$

A.3 CO Numerical Example

The complete collaborative optimization formulation for the numerical example described above can be seen in Figure A-1. The system level optimizer uses global variables identified by Z with a subscript that identifies which local variable or coupling variable it is optimizing. This is done to make the architecture clearer. The system level constraints are simply the J values from discipline one and discipline two being set equal to zero. The subspace optimization routines are set with the task of matching the global Z values while satisfying the local constraints and analysis. The starting global Z values, given next, and are the same as with both the MDF and IDF methods.

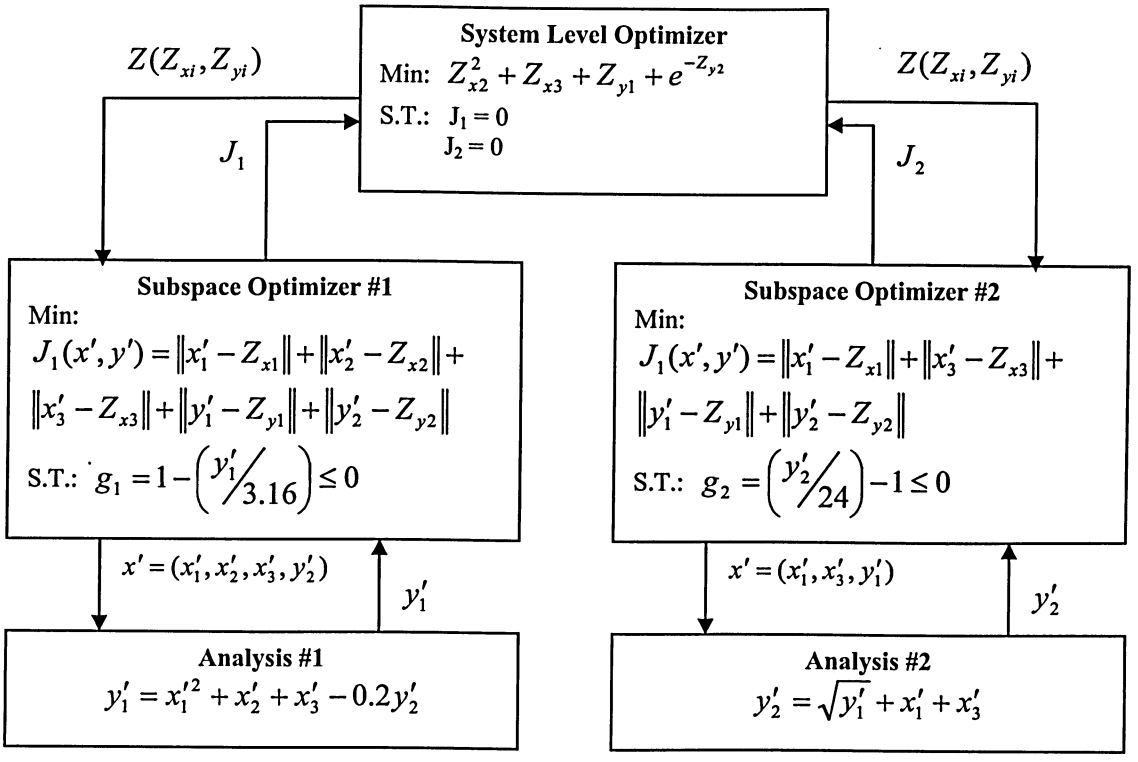


Figure A-1: Example of CO problem formulation

$$Z_{x1} = 1, \quad Z_{x2} = 5, \quad Z_{x3} = 2$$
$$Z_{y1} = 10, \quad Z_{y2} = 4$$

The subspace optimizers must now independently match these values that the system level optimizer has set out. For the first iteration, discipline number one yielded design variables values of,

$$x_1' = 1.7715, \quad x_2' = 5.2178, \quad x_3' = 2.2177$$
$$y_1' = 7.0, \quad y_2' = 3.9565$$

with a J_1 value of,

$$J_1 = 0.7393$$

For the first iteration, discipline number two yielded design variable values of,

$$x_1' = 0.2852, \quad x_2' = 5.0, \quad x_3' = 1.2852$$
$$y_1' = 9.8863, \quad y_2' = 4.0$$

with a J_2 value of,

$$J_2 = 1.5455$$

The next step is to pass the values of J_1 and J_2 to the system level optimizer to see if the interdisciplinary compatibility constraints are satisfied. For this iteration interdisciplinary compatibility constraints are clearly not satisfied and another iteration must be done and the global optimizer must choose a new Z vector. This process must continue until both values of J_1 and J_2 are zero (10^{-12}). Upon convergence both discipline one and discipline two will yield the same design variable values.

The entire optimization process achieved convergence after approximately 288 function evaluations. However, at 50 function evaluations there is very little change of the design variables. The extra function evaluations are used to satisfy the equality constraint to 10^{-12}. The convergence of the problem can be seen below in Figure A-2. Figure A-2 a. shows the complete convergence of the optimization. Figure A-2 b. is a zoomed shot of the convergence to function evaluation 150.

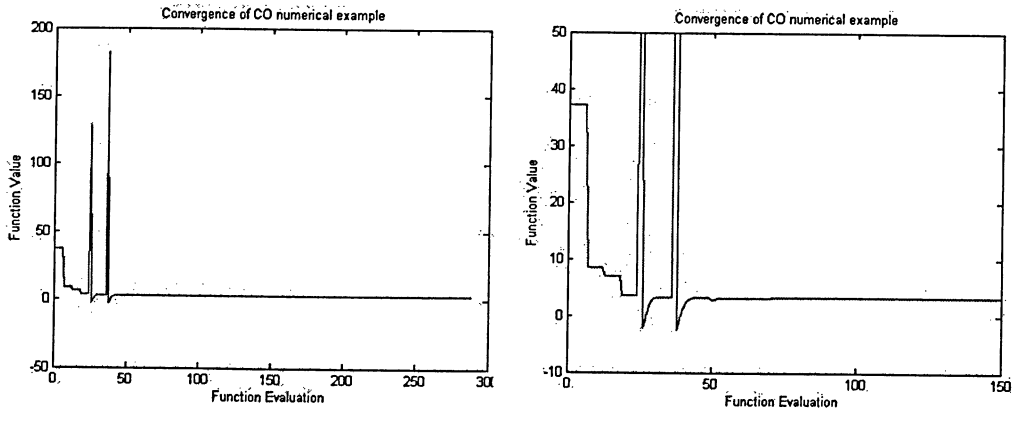


Figure A-2: a. Convergence of numerical example. b. Zoomed convergence

From these figures it is clear that the value of the function value changes very little after function evaluation 50. Overall, the optimization converged in 34 system level optimization iterations. The optimization converged with global design variable values of,

$$Z_{x1} = 1.97732, \qquad Z_{x2} = 0.00129, \qquad Z_{x3} = 0$$
$$Z_{y1} = 3.16, \qquad Z_{y2} = 3.7553$$

which is close to the values that were obtained by Perez et al. [23] and the MDF and IDF methods. The only difference was that the value of x_2 did not completely converge to zero.

Appendix B

MATLAB Wrapper

In order to facilitate the use of several different software packages a wrapper was created within MATLAB that would be able handle all of the input and output files that are created by GAMBIT, FLUENT, and ANSYS. The MATLAB wrapper can be broken down into three main modules that control the CFD and FEM model geometry and the CFD and FEM interactions. The first module is able to write a GAMBIT journal file code to create the geometry. It is also capable of automatically creating a grid around the wing while taking into consideration grid continuity and grid skewness. The second module is capable of writing FLUENT journal code that changes the solving parameters based on the user inputs. It controls the CFD while monitoring the residuals and the solution convergence. Finally, it also exports the necessary pressures at user defined intervals along the wing for use in the FEA. The last module controls the ANSYS inputs and outputs. It creates an ANSYS input file that builds a wing with N number of ribs and M number of stringers and assigns the proper element type and thickness. It is also used to apply the aerodynamic forces to the structural model and perform the automatic meshing of the wing. All of these modules are linked via the MATLAB wrapper that calls these modules in a specific sequence. The sequence of module calls performs the static aeroelastic analysis by coupling the aerodynamic load and structural displacement data. The following section outlines how the wrapper was created, how each software

package uses some of the data given to it, and gives detailed information about the flow of data. Overall, this section provides details concerning the use of all of the high-fidelity software packages used for this research.

B.1 GAMBIT Module

B.1.1 Interfacing with GAMBIT

The GAMBIT module was created to automatically generate the 3D wing geometry and to perform the grid generation around the wing. The ability to automatically generate a new 3D wing shape based on the optimizer input was crucial for a successful optimization. In order to interface with GAMBIT a journal file needed to be created that contained a listing of commands that will be used to create and mesh the geometry. Obtaining the commands that are required to do a specific job is easy to obtain in GAMBIT. When using the GUI in GAMBIT all of the equivalent text commands are recorded in a journal file. Once the file has been finished using the GUI, the journal file can be run to repeat the process if desired. Sometimes the user will be required to manually type the commands into a text file. This is because some commands are not available from the GUI. In addition to this, the journal must be able to read in arrays of data. This is critical for reading in airfoil coordinates and creating the wing shape.

B.1.2 Journal Code Example Command

The following example provides a brief introduction to a few of the commands that can be used to manipulate data using the GAMBIT journal file. To start, it is assumed that a data array has already been inputted into GAMBIT in the variable *$fc* that contains all the data points needed to create an airfoil. The next step is to create all of the vertices that will make the airfoil shape. This can be done by using a *"do"* loop within the GAMBIT journal. Note that this needs to be entered manually and normally does not use the GUI. The set of commands to call the *"do"* loop and create all the airfoil vertices is as follows.

```
/do para "$x" init 1 cond ($x .le. $num_of_vert) incr (1)
/vertex create coordinates $fc[1,$x] $fc[2,$x] $fc[3,$x]
/enddo
```

The variable '*$x*' is a numerical counter that is used to count indices. The condition given by the command '*cond*' uses a FORTRAN symbol of '*le*' or less than/equal that determines when the loop should exit. The command '*incr*' sets the amount that *$x* should be incremented during each loop. The second line is the command that creates vertices in GAMBIT, which can be easily obtained from any GAMBIT user manual. Finally, the last line simply ends the *"do"* loop. This is just an example of a typical command and many more commands contained in the GAMBIT environment.

B.1.3 Calling GAMBIT from MATLAB

Once the entire GAMBIT file has been created and is error free, it can be used to create the desired model nearly instantly. GAMBIT has an excellent batch mode that runs completely in the background on either Windows or UNIX work stations. To call GAMBIT in batch mode in the UNIX environment the following command should be used.

/path/gambit -id sessionname -inp input.jou > output.out

Similarly, the batch mode can be called in Windows using:

c:\path\gambit -id sessionname -inp input.jou > output.out

The path identifies the location of the GAMBIT executable file on the local system. The term '*-id sessionname*' identifies the name of the current session. The term '*-inp input.jou*' forces GAMBIT to run in batch mode and calls the identified journal file. Finally, '*> output.out*' contains any errors may occur during the current run. The output should be empty at all times, and if it is not then the journal file was not made correctly, and changes need to be made.

When GAMBIT is required to be called from MATLAB, it is necessary to use the MATLAB '*system*' command. To call GAMBIT in batch mode from the Windows MATLAB environment the following command must be used:

system('c:\path\gambit -id sessionname -inp input.jou > output.out');

Once this command has been implemented MATLAB will be locked until the current '*system*' command has been completed. If GAMBIT is being used in a default setting the entire journal file will be echoed back the MATLAB workspace. To prevent this from happening the default setting for GUI/transcript must be changed from '1' to '-1' within the GAMBIT environment.

B.2 FLUENT Module

B.2.1 Interfacing with FLUENT

The FLUENT module was created to solve the CFD and to output important information about the grid and the pressure distribution around the wing. Because a lot of the information used in the CFD solver can be changed by the user, a MATLAB function was made that can write journal files to implement theses changes. For example, the user can change the number of span wise stations to output the pressure distribution or even angle of attack. To output these pressures requires a different journal code each time a new station is needed to be outputted. In addition, the journal file also controls the output of the CFD grid for use in the mesh mapping function. In order to make the FLUENT journal file, the journal commands must be entered in a specific order. Even one single blank line can cause an error and make the file crash. To find the required journal commands is quite easy because FLUENT can be used in the GUI mode or the text mode (typing the required commands) simultaneously. When using the text mode all of the options are given and the desired command can be typed and the results seen immediately. Another method of obtaining the journal command is to use the function within FLUENT that can record the mouse clicks in the GUI mode and return the equivalent journal file.

B.2.2 Journal Code Example Command

The following is an example command that is used within this research to create a span wise station along the wing to export the pressure. For this example, it is assumed that the span of the wing extends in the z-direction and the top portion on the wing is identified in FLUENT by '*wingtop*'. To create the span wise station the '*iso-surface*'

command must be used to identify the location of the station. The following is an example of the required commands,

iso-surface
z-coordinate
z=20.37top
wingtop
20.37

The command '*iso-surface*' opens the station creation function. The '*z-coordinate*' identifies that the station should be made in the z direction, '*z=20.37top*' is the name of the station that is being made, and '*wingtop*' identifies where the data should be taken from the local station when exporting. Finally, the distance the station is along the span is given by '*20.37*'. To export the pressures to a file from this station the following commands can be used.

export
ascii
pressure.out
z=20.37top

The command '*export*' enters the exporting function and '*ascii*' prepares FLUENT to export an ASCII text file. The file name that the pressure should exported to is called '*pressure.out*', with the pressure data coming from the previously created station '*z=20.37top*'.

B.2.3 Calling FLUENT from MATLAB

FLUENT can be called in a batch mode and solved while in the background using a journal file to control the solution process. The commands to call FLUENT in batch mode are very similar between Windows and UNIX environments. When calling FLUENT in batch mode while in UNIX, the FLUENT program runs entirely in the background and when FLUENT is called in Windows the program is visible and solution can be viewed. Each system has its own pros and cons. The command to call FLUENT in batch mode while in the UNIX environment is,

/path/fluent 3d -g < journalfile.jou > output.out

The Windows equivalent command is,

\path\fluent 3d -g -i journalfile.jou -o output.out

It can be seen that '<' and '*-i*' as well as '>' and '*-o*' are equivalent commands. The command '*journalfile.jou*' is the file that should contain all of the required commands to solve the CFD equations and '*output.out*' is where echoed data from FLUENT is dumped. If FLUENT needs to be called from MATLAB the following command should used:

system('\path\fluent 3d -g -i journalfile.jou -o output.out');

When this command is implemented, FLUENT will be called and the current problem solved. However, if there is an error in either the mesh file or the journal file, FLUENT will report an error and will stop the solution. If this happens, MATLAB will not regain control until FLUENT has been exited manually.

B.3 ANSYS Module

B.3.1 Interfacing with ANSYS

In order to successfully perform optimization using a high-fidelity tool such as ANSYS it is necessary to create an interface to control the flow of information. Within ANSYS there exists a command language called ANSYS parametric design language (APDL) that can perform any task that the GUI can. Using this language, it is possible to control the geometry creation, meshing, solving, and post-processing of the analysis. The following section details a few of the commands and processes that were used during the aeroelastic analysis.

ANSYS can be used in two different modes. The first is the GUI that gives the user the ability to see the changes that are being made. The second mode is the ANSYS batch mode that reads a FORTRAN based text file that performs all of the necessary commands in the background. This is known as the APDL. In order to use the batch mode the user is required to create a text file that controls the model creation. Creation of this file is not a trivial task, and for complex analysis and geometries APDL should only be used by an experienced ANSYS user.

When using ANSYS in the graphical mode one can simply click on the commands that the user wishes to perform. Each of these commands is stored within a log file that the user can read. Performing a task in the GUI and referring to the log file

can reveal the text command that can do the equivalent when ANSYS is run in batch mode. For example, if the user wishes to create a key point with coordinates at (1, 3, 4) the command K, ,1,3,4 should be entered.

B.3.2 APDL Example Command

The following is an example of how to read in a series of points into an array in ANSYS. For this example a list of coordinates that define the shape of an airfoil is used. If all of the airfoil coordinates are stored within a plain text file it is necessary to input these points into ANSYS in order to create the airfoil. The first command that should be implemented is to define a 2-D array that is of the required size. This is done using the following command.

**DIM,arrayname,,numrows,numcols*

The next step is to read the airfoil coordinates into the defined array. This is done using the '*vread*' command. Using this command requires three lines of text.

**VREAD,arrayname(1,1),filename.fileextension,,JIK,numcols,numrows*
(3F16.9)
**stat,arrayname(1,1)*

The first line reads the data from the file '*filename.fileextension*' that has the size of '*numcols*' and '*numrows*'. The second line defines the format that should be used when the file is read. The number before the *F* defines number of pieces of data in each row. The numbers after the *F* defines the number of digits before and after the decimal place respectively. For a specific listing of commands and how they are used please refer to Ref. [86] or Appendix C to view the aeroelastic ANSYS input file.

B.3.3 Calling ANSYS from MATLAB

After the entire input text file is created it can be used to run ANSYS in batch mode. Running ANSYS in batch mode is a simple task if the input file has been created properly. For both UNIX and Windows based systems the commands are very similar. The UNIX command is the following,

/path/ansys_executable –b –p ansysproduct –i inputfile –o outputfile

The Windows equivalent command is:

C:\path\ansys_ executable –b –p ansysproduct –i input file –o outputfile

the path is where the executable is found within the designated system. The '*–b*' indicates that ANSYS should be called in batch mode. The '*–p ansysproduct*' indicates which product should be used when ANSYS is loaded. The '*–i*' and '*–o*' are the input and output files respectively. The input file should be the input text file that contains all of the required ANSYS commands for the analysis. The output file will contain all of the information that is usually echoed back via the command screen. This will also include any errors that are found within the current analysis.

Sometimes it is necessary to call ANSYS directly from MATLAB so updated parameters can be used to update the geometries in the batch file. To call ANSYS from MATLAB the above batch mode commands can be used along with the built in MATLAB function '*system*'. The command should be entered as follows.

system('C:\path\ansys_executable –b –p ansysproduct –i input file –o outputfile');

When this command is implemented, MATLAB will pause while ANSYS runs in the background. Once ANSYS is finished the control switches back to MATLAB so the remaining script can run. Running ANSYS from MATLAB can be helpful when many runs of one simulation are required. For example, the aeroelastic wing that is being considered here has many geometrical input requirements that change from one function evaluation to the next. One function evaluation may require a sweep and taper combination of 25 degrees and 0.4 respectively and the next may require 5 degrees and 0.8. Overall, the ANSYS input file must be robust enough to handle all of the different changes that the optimizer can possibly require.

B.4 MATLAB Wrapper Data Flow

The entire MATLAB wrapper consists of a many input and output files that are used to control the aeroelastic coupling process and optimization. This section details

how each piece of data interacts with one another. These interactions can be seen in the flow chart contained in the following pages. The flow chart contains specific symbols and colors that represent certain events and files. The meaning of the symbols that are used in the flow chart can be seen in Figure B-1.

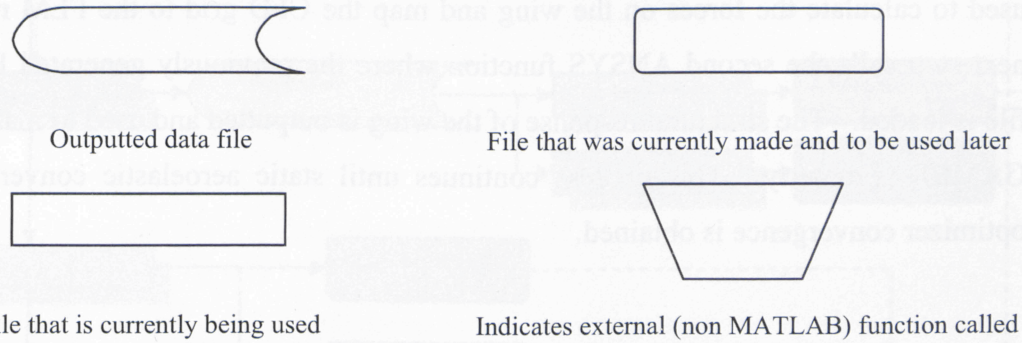


Figure B-1: Symbol identification for MATLAB wrapper flow chart

The colours that are used in the flow chart represent which program is to use or is using the current file. In addition, some sets of symbols are surrounded by a lightly coloured box that represents which module the files are used in. The colours and their meaning can be seen below in Figure B-2.

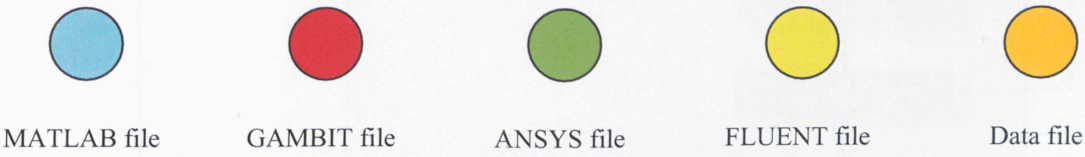
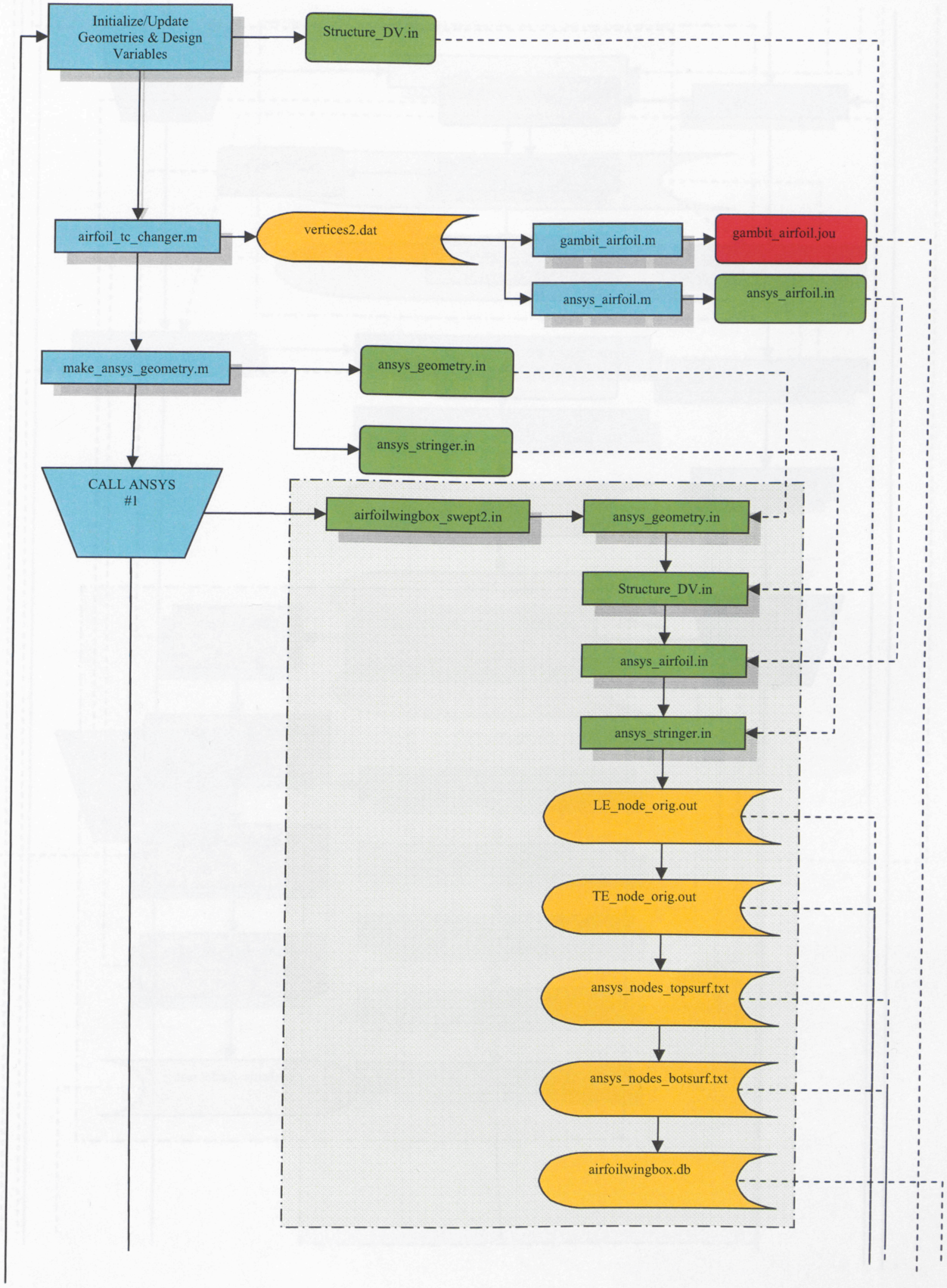


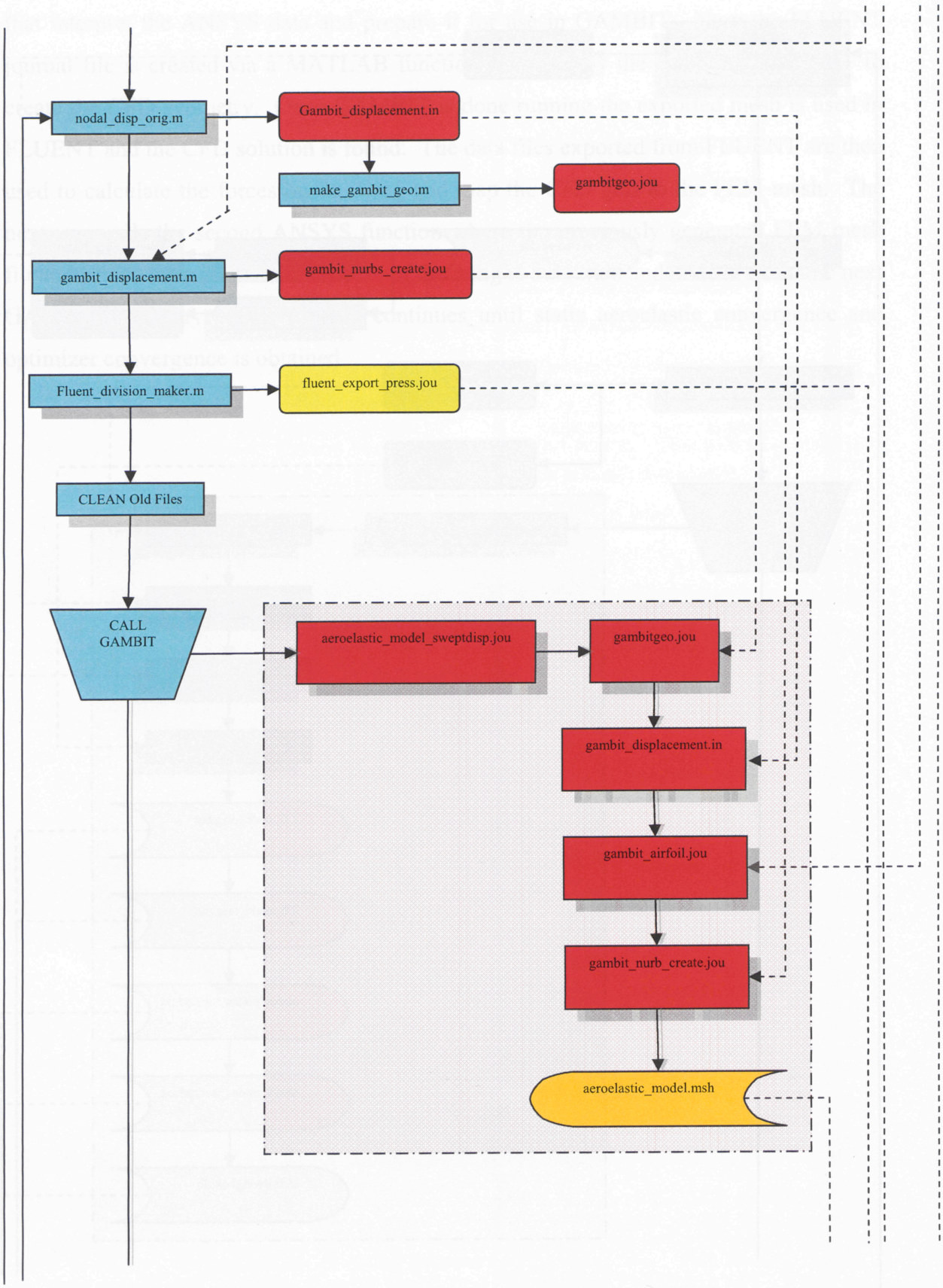
Figure B-2: Colour identification for MATLAB wrapper flow chart

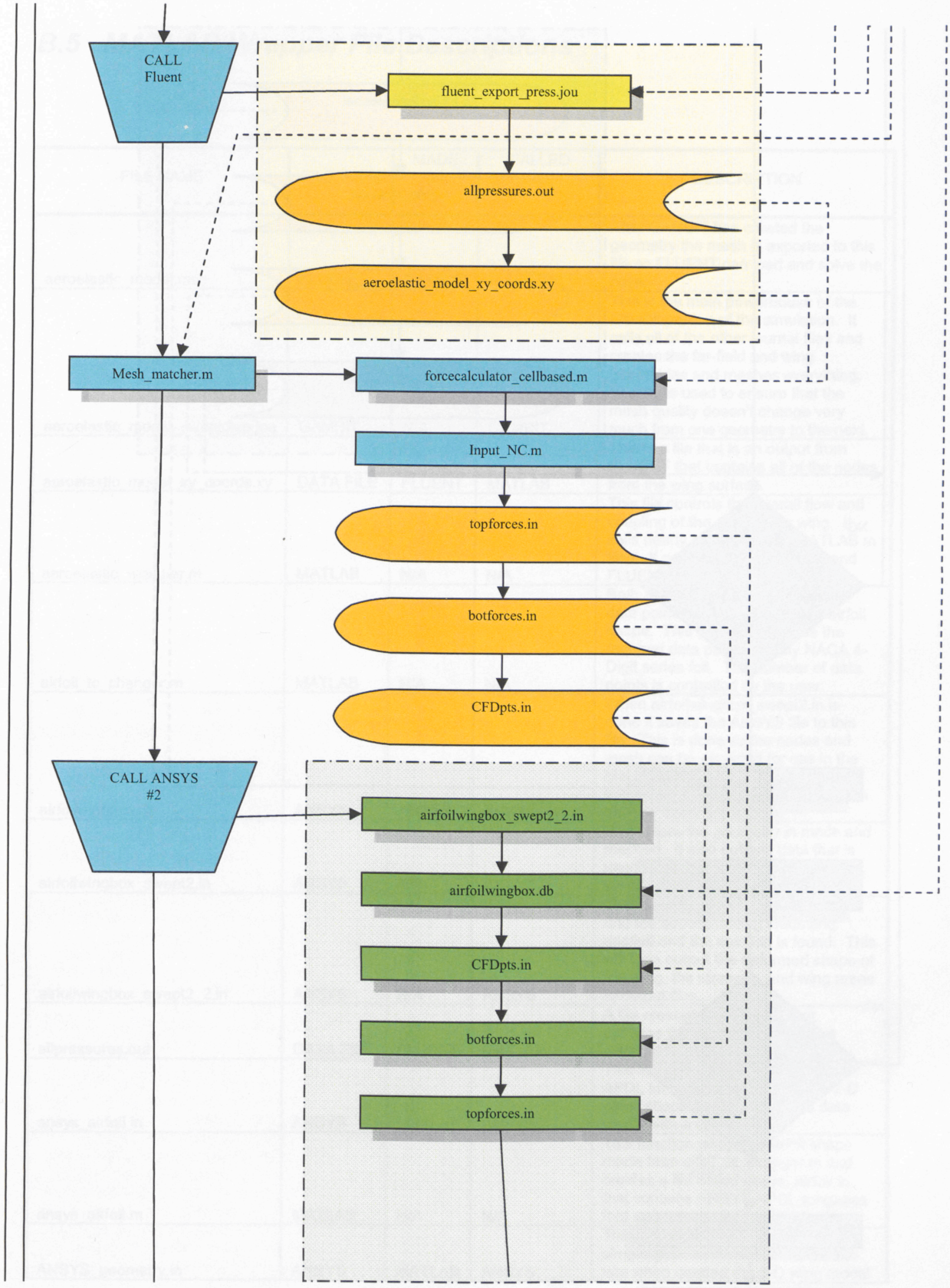
In addition to the colours and symbols there are also solid and dashed lines that show the flow of the data. The solid lines are use to represent the direct flow of data, and the dashed lines are used to make the tracing of file creation and usage easier.

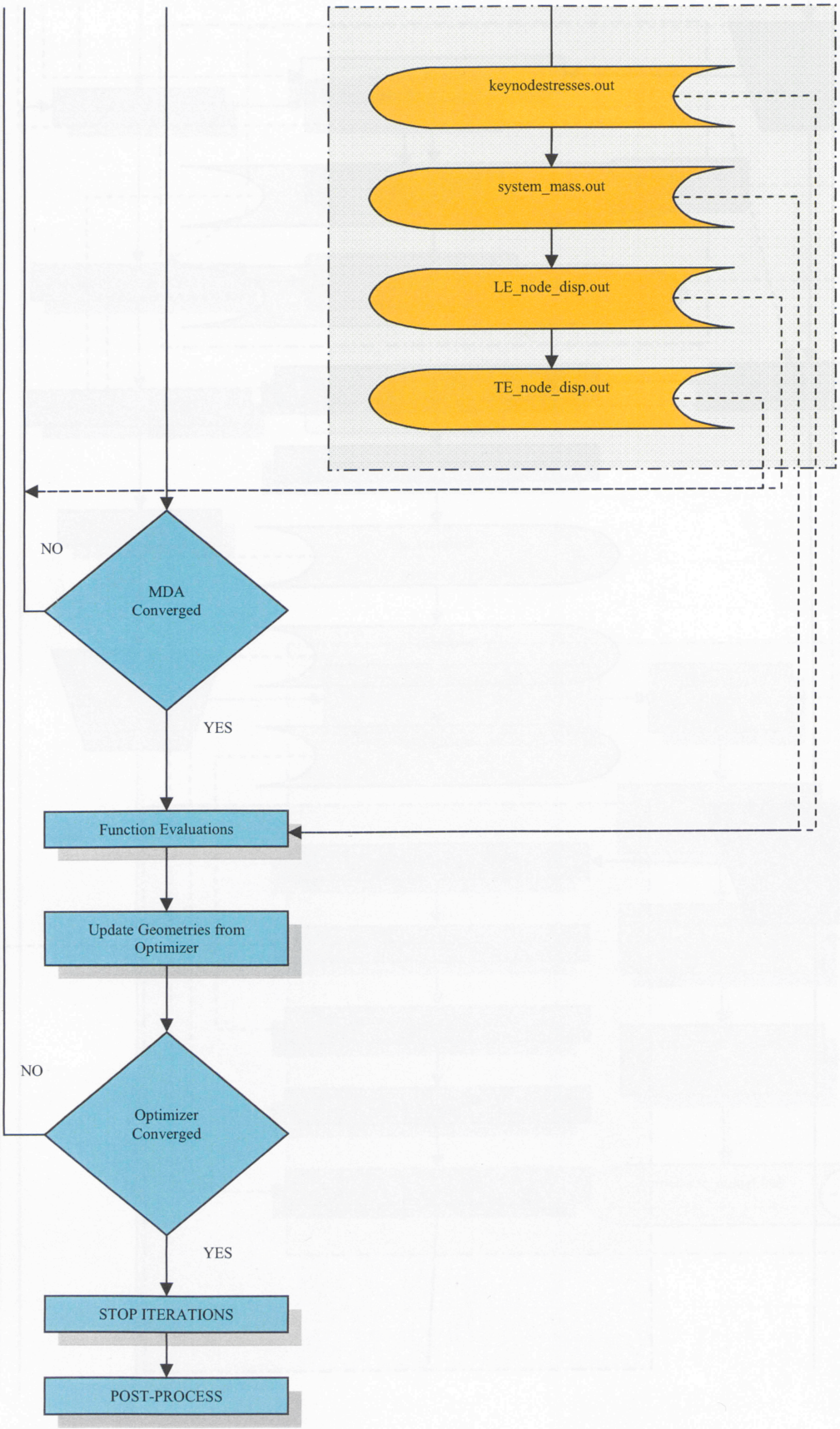
The first thing that the wrapper does is to initialize all of the geometrical design variables that are used within the analysis. Updating of the geometrical design variables is usually done by the optimizer. The next step is to create the vertices that make up the airfoil shape and then format the file so that the geometry is compatible with both GAMBIT and ANSYS. The function that creates all of the APDL code that makes the FEM model in ANSYS is called and then followed by ANSYS itself. The outputs from this first ANSYS call are required because they are used to create the wing geometry in GAMBIT. The GAMBIT geometry is now made using a series of MATLAB functions

that interpret the ANSYS data and prepare it for use in GAMBIT. Now the FLUENT journal file is created via a MATLAB function followed by the calling of GAMBIT to create the CFD geometry. Once GAMBIT is done running the exported mesh is used in FLUENT and the CFD solution is found. The data files exported from FLUENT are then used to calculate the forces on the wing and map the CFD grid to the FEM mesh. The next step calls the second ANSYS function where the previously generated FEM mesh file is loaded. The structural response of the wing is outputted and used to make the next GAMBIT geometry. This process continues until static aeroelastic convergence and optimizer convergence is obtained.


Initialize/Update Geometries & Design Variables
Structure_DV.in
airfoil_tc_changer.m
vertices2.dat
gambit_airfoil.m
gambit_airfoil.jou
ansys_airfoil.m
ansys_airfoil.in
make_ansys_geometry.m
ansys_geometry.in
ansys_stringer.in
CALL ANSYS #1
airfoilwingbox_swept2.in
ansys_geometry.in
Structure_DV.in
ansys_airfoil.in
ansys_stringer.in
LE_node_orig.out
TE_node_orig.out
ansys_nodes_topsurf.txt
ansys_nodes_botsurf.txt
airfoilwingbox.db


nodal_disp_orig.m
Gambit_displacement.in
make_gambit_geo.m
gambitgeo.jou
gambit_displacement.m
gambit_nurbs_create.jou
Fluent_division_maker.m
fluent_export_press.jou
CLEAN Old Files
CALL GAMBIT
aeroelastic_model_sweptdisp.jou
gambitgeo.jou
gambit_displacement.in
gambit_airfoil.jou
gambit_nurb_create.jou
aeroelastic_model.msh


CALL
Fluent
fluent_export_press.jou
allpressures.out
aeroelastic_model_xy_coords.xy
Mesh_matcher.m
forcecalculator_cellbased.m
Input_NC.m
topforces.in
botforces.in
CFDpts.in
CALL ANSYS
#2
airfoilwingbox_swept2_2.in
airfoilwingbox.db
CFDpts.in
botforces.in
topforces.in


keynodestresses.out
system_mass.out
LE_node_disp.out
TE_node_disp.out
NO
MDA Converged
YES
Function Evaluations
Update Geometries from Optimizer
NO
Optimizer Converged
YES
STOP ITERATIONS
POST-PROCESS

B.5 MATLAB Wrapper File Descriptions

FILE NAME	FILE TYPE	MADE IN	CALLED FROM	DESCRIPTION
aeroelastic_model.msh	FLUENT	GAMBIT	FLUENT	After GAMBIT has created the geometry the mesh is exported to this file so FLUENT can load and solve the case.
aeroelastic_model_sweptdisp.jou	GAMBIT	N/A	GAMBIT	This is the main powerhouse of the Gambit portion of the simulation. It calls all of the other journal files and creates the far-field and wing geometries and meshes everything. Scaling is used to ensure that the mesh quality doesn't change very much from one geometry to the next.
aeroelastic_model_xy_coords.xy	DATA FILE	FLUENT	MATLAB	This is a file that is an output from FLUENT that contains all of the nodes from the wing surface.
aeroelastic_wrapper.m	MATLAB	N/A	N/A	This file controls the overall flow and coupling of the aeroelastic wing. It calls nearly all of the main MATLAB m files. It calls ANSYS, GAMBIT, and FLUENT.
airfoil_tc_changer.m	MATLAB	N/A	N/A	Both ANSYS and GAMBIT require data points to make the overall airfoil shape. This function provides the required data points for any NACA 4-Digit series foil. The number of data points is controlled by the user.
airfoilwingbox.db	ANSYS	ANSYS	ANSYS	When airfoilwingbox_swept2.in is done it saves the ANSYS file to this file. This is done so the nodes and mesh can be exported for use in the Mesh_matcher.m. This file is then loaded by airfoilwingbox_swept2_2.in and the loads are applied.
airfoilwingbox_swept2.in	ANSYS	N/A	ANSYS	This where the geometry in made and meshed. It also outputs data that is used for Mesh_matching.m. It saves the session to airfoilwingbox.db
airfoilwingbox_swept2_2.in	ANSYS	N/A	ANSYS	The file airfoilwingbox.db is loaded and the corresponding loads and applied and the solution is found. This will then output the deformed shape of the wing, the stresses, and wing areas to output files.
allpressures.out	DATA FILE	FLUENT	MATLAB	A file created by FLUENT that contains the discretized pressure distribution along the wing.
ansys_airfoil.in	ANSYS	MATLAB	ANSYS	This is the file that contains ANSYS APDL language that creates the 2-D airfoil shape and connects the data points with a spline curve.
ansys_airfoil.m	MATLAB	N/A	N/A	This function takes the airfoil shape made from aifoil_tc_changer.m and creates a file called ansys_airfoil.in that contains ANSYS APDL language that creates any 2-D airfoil shape.
ANSYS_geometry.in	ANSYS	MATLAB	ANSYS	This is a small file that holds the simple geometric data for ANSYS to use when created the 3-D wing model

ANSYS_LocThick.in	DATA FILE	MATLAB	MATLAB	This is a file created by MATLAB that has each ANSYS real constant defined and the corresponding element thickness. This is used in combination with wing_areas.txt to calculate the mass of the wing.
ansys_nodes_botsurf.txt	DATA FILE	ANSYS	MATLAB	This is an ANSYS file that contains all of the nodes from the meshed FEM model bottom surface of the wing.
ansys_nodes_topsurf.txt	DATA FILE	ANSYS	MATLAB	This is an ANSYS file that contains all of the nodes from the meshed FEM model top surface of the wing.
ansys_stringer.in	ANSYS	MATLAB	ANSYS	This the main ANSYS file that creates the wing shape and creates the necessary number of ribs and stringers. This is the most delicate function in the entire simulation.
botforces.in	DATA FILE	MATLAB	ANSYS	A file used by ANSYS that contains a node number and a corresponding x any y force that is applied to the bottom of the wing.
CFDpts.in	DATA FILE	MATLAB	ANSYS	This is a file that ANSYS reads so it knows how big to define the force martices.
Fluent_division_maker3.m	MATLAB	N/A	N/A	This is version 3 of this file. It writes a FLUENT journal file that sets up the problem. All aspects of the FLUENT solution are controlled from this file. It writes the FLUENT journal file fluent_export_press.jou
fluent_export_press.jou	FLUENT	MATLAB	FLUENT	This file loads the mesh created by GAMBIT sets up the initial conditions and solves the wing until convergence. It saves all coefficient data and outputs the pressure distribution along the wing to allpressures.out. It also outputs the locations of all of the nodes along the wing in aeroelastic_model_xy_coords.xy.
forcecalculator_cellbased.m	MATLAB	N/A	MATLAB	This file takes all of the cell centred pressures from allpressures.out and uses the calculated element areas to convert the pressure for forces. Then uses the cell normals to convert the forces into x and y forces for application to the FEM model,
gambit_foil.jou	GAMBIT	MATLAB	GAMBIT	This is a journal file for Gambit that creates a 2-D airfoil shape.
gambit_airfoil.m	MATLAB	N/A	N/A	This function takes the airfoil shape made from airfoil_tc_changer.m and creates two files called gambit_foil.jou and gambit_foil.fix.jou. Each file contains Gambit journal file commands that make a smooth 2-D airfoil in Gambit.
Gambit_displacement.in	GAMBIT	MATLAB	GAMBIT	This file contains Gambit journal code that loads the deformed shape of the wing into Gambit.
gambit_displacement.m	MATLAB	N/A	MATLAB	This file uses the data loaded by Gambit_displacement.in to create the journal file gambit_nurb_create.jou that can make vertices and draw a nurbs line to form the entire deformed LE and TE of the wing. The user has the option of changing the number of points to use to define the nurbs line.

gambit_foil_fix.jou	GAMBIT	MATLAB	GAMBIT	This is a journal file for Gambit that connects the C-grid to the foil. This is necessary because depending the on the number of vertices used the LE and TE vertices are labeled differently. This takes those possible changes in to account.
gambit_nurb_create.jou	GAMBIT	MATLAB	GAMBIT	This is a journal file for Gambit that draws the NURBS line through data points from the root to tip that defines the deformed wing shape.
gambitgeo.jou	GAMBIT	MATLAB	GAMBIT	This file defines all of the geometric variables for use in Gambit.
Input_NC.m	MATLAB	N/A	MATLAB	Controls the input of several pieces of data. Outputting data without headers in ANSYS forces ANSYS to output data for all elements or nodes. This m file takes that data and returns only data that is required. (i.e. Only nodes from the LE or TE)
keynodestresses.out	DATA FILE	ANSYS	MATLAB	Contains Von-Mises stresses from all of the elements in the FEM model.
LE_node_disp.out	DATA FILE	ANSYS	MATLAB	This is a vector from ANSYS that defines a displaced line from the root to the tip for the LE.
LE_node_orig.out	DATA FILE	ANSYS	MATLAB	This is a vector from ANSYS that defines an undeformed line from the root to the tip for the LE.
make_ansys_geo2.m	MATLAB	N/A	MATLAB	This file used to create the ANSYS APDL code that will create a 3-D wing in the shape defined by the user. The user has control of all aspects of the wing. It also defined the corresponding real constants for each individual areas.
make_gambit_geo.m	MATLAB	NA	MATLAB	This creates the file gambitgeo.jou that defines some geometric variables for use in Gambit.
Mesh_matcher.m	MATLAB	N/A	MATLAB	The output from forcecalculator_cellbased.m is the forces for the top and bottom of the wing in separate matrices. The mesh matcher then will find the closest FEM model node and assigns a corresponding x and y force. All of this is then save to two files for the top and bottom of the wing call topforces.in and botforces.in
nodal_disp_orig.m	MATLAB	N/A	MATLAB	This file reads in data files outputted from ANSYS that define the shape of the LE and TE of the deformed flexible wing. This writes the Gambit journal file gambit_displacement.in that loads all of the points into Gambit.
Structure_DV.in	DATA FILE	MATLAB	ANSYS	A small file that contains all of the root thicknesses that are chosen by the user (or optimizer).
TE_node_disp.out	DATA FILE	ANSYS	MATLAB	This is a vector from ANSYS that defines a displaced line from the root to the tip for the TE.
TE_node_orig.out	DATA FILE	ANSYS	MATLAB	This is a vector from ANSYS that defines an undeformed line from the root to the tip for the TE.
topforces.in	DATA FILE	MATLAB	ANSYS	A file used by ANSYS that contains a node number and a corresponding x any y force that is applied to the top of the wing.

vertices2.dat	DATA FILE	MATLAB	MATLAB	This is the file that contains the coordinates of the 2-D airfoil shape.
wing_areas.txt	DATA FILE	ANSYS	MATLAB	All of the areas in ANSYS and their corresponding area (in m^2) are output here. This is used to calculate the mass of the wing.

Appendix C

Example ANSYS APDL Code

```
!  ****************************************************************
!  *****************   MASc High Fidelity MDO       ***************
!  ****                    By: Brian Leonard                    ****
!  ****                   ANYSY INPUT FILE #2                   ****
!
!  Version                    Changes                      Date
! ------------------------------------------------------------------
!  1.0              Initial Full Working Version      AUG  5, 2005
!  2.0              Changed the stress outputting     AUG 25, 2005
!  3.0              Changed the displacement output
!                   Changed the force application     OCT  2, 2005
!
!  ****************************************************************
!
!  INPUTS:      - airfoilwingbox.db: ANSYS database that contains
!                                    the meshed wing
!               -         CFDpts.in: file that contains the number
!                                    CFD forces to apply to the wing
!               -      topforces.in: A file that contains a listing
!                                    of all nodes and corresponding
!                                    forces for top of wing
!               -      botforces.in: A file that contains a listing
!                                    of all nodes and corresponding
!                                    forces for bottom of wing
!
! OUTPUTS:    - keynodestresses.out: Contains all the nodes and all
!                                    of the stresses
!             -       wing_area.txt: All of the wing areas
!             -    LE_node_disp.out: A listing of all the deflected
!                                    nodes on the leading edge
!             -    TE_node_disp.out: A listing of all the deflected
!                                    nodes on the trailing edge
!
!
! THIS FUNCTION LOADS THE MESHED WING AND APPLIES ALL OF THE FORCES
! TO EACH NODE ON THE MODEL.  THE BOUNDARY CONDITIONS ARE APPLIED
```

```
! AND THE MODEL IS SOLVED.  IMPORTANT IMFORMATION IS OUTPUTTED
! TO BE POST-PROCESSED IN MATLAB.
!
!
!  ************************************************************
/BATCH  !starts batch mode

RESUME,'airfoilwingbox','db','',,   !Loads the airfoil mesh

!this reads a file that contains the number
!cfd pts to be applied to the upper and
!lower surfaces
/INPUT,'CFDpts','in','',1,0
/VIEW,1,1
!,1,1
/ANG,1
/REP,FAST
/AUTO,1
/REP,FAST

!Reads in the forces and the corresponding nodes
!and stores them in the array TF
*DIM,TF,,CFDpts,3
*VREAD,TF(1,1),topforces,in,,JIK,3,CFDpts
(3F16.3)
*stat,TF(1,1)

!Reads in the forces and the corresponding nodes
!and stores them in the array BF
*DIM,BF,,CFDpts,3
*VREAD,BF(1,1),botforces,in,,JIK,3,CFDpts
(3F16.3)
*stat,BF(1,1)

/SOLU    !starts the solution portions of ANSYS

!Applies all the forces to the nodes at specific intervals
*do,jj,1,CFDpts
 /GO
   FLST,2,1,1,ORDE,1
   FITEM,2,TF(jj,1)
   F,P51X,FX,TF(jj,2)
*enddo

*do,jj,1,CFDpts
   FLST,2,1,1,ORDE,1
   FITEM,2,TF(jj,1)
   F,P51X,FY,TF(jj,3)
*enddo

*do,jj,1,CFDpts
 /GO
   FLST,2,1,1,ORDE,1
   FITEM,2,BF(jj,1)
   F,P51X,FX,BF(jj,2)
*enddo
```

```
*do,jj,1,CFDpts
   FLST,2,1,1,ORDE,1
   FITEM,2,BF(jj,1)
   F,P51X,FY,BF(jj,3)
*enddo


!Cantilevers the root
ASEL,S,LOC,Z,0
DA,ALL,ALL,0

!Solves the problem
/STATUS,SOLU
SOLVE

!Starts the post-processing of the model
/POST1

!Selects all the nodes to find the max stress
!and outputs them to an output file
ALLSEL,ALL
*VGET,stress,NODE, ,S,EQV, , ,2
*CREATE,ansuitmp
*CFOPEN,'keynodestresses','out',' '
*VWRITE,stress(1), , , , , ,
(1F15.5)
*CFCLOS
*END
/INPUT,ansuitmp

/prep7    !Enters pre-processor
ASUM      !Calculates all area information
*VGET,areas,AREA, ,AREA, , ,2
*VGET,areasthick,AREA, ,ATTR,REAL, , ,2


/POST1    !Enters post-processor

!Outputs all of the wing area and the corresponding
!thickness to an output file
*CREATE,ansuitmp
*CFOPEN,'wing_areas','txt',' '
*VWRITE,areas(1),areasthick(1), , , , , ,
(2F15.10)
*CFCLOS
*END
/INPUT,ansuitmp


ALLSEL,ALL   !Selects all the entities


!Selects the LE and TE components then the nodes
CMSEL,S,LE
NSLL,R,1
```

```
*VGET,S,NODE,1,NSEL,, , ,2    !Gets the selection status
*VGET,xl,NODE,1,U,X, , ,2     !Gets the displacements in the x
*VGET,yl,NODE,1,U,Y, , ,2     !Gets the displacements in the y
*VGET,zl,NODE,1,U,Z, , ,2     !Gets the displacements in the z


!Outputs all LE edge deflected nodes
*CREATE,ansuitmp
*CFOPEN,'LE_node_disp','out',' '
*VWRITE,S(1),xl(1), yl(1), zl(1), , , , , ,
(4F20.10)
*CFCLOS
*END
/INPUT,ansuitmp


ALLSEL,ALL    !Selects all the entities
CMSEL,S,TE
NSLL,R,1

*VGET,S,NODE,1,NSEL,, , ,2    !Gets the selection status
*VGET,xl,NODE,1,U,X, , ,2     !Gets the displacements in the x
*VGET,yl,NODE,1,U,Y, , ,2     !Gets the displacements in the y
*VGET,zl,NODE,1,U,Z, , ,2     !Gets the displacements in the z


!Outputs all LE edge deflected nodes
*CREATE,ansuitmp
*CFOPEN,'TE_node_disp','out',' '
*VWRITE,S(1),xl(1), yl(1), zl(1), , , , , ,
(4F20.10)
*CFCLOS
*END
/INPUT,ansuitmp

!Saves the results and exits
SAVE,'airfoilwingbox_results','db','',
```

(5) 20-44-44