A COUPLED-ADJOINT METHOD FOR HIGH-FIDELITY AERO-STRUCTURAL OPTIMIZATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Joaquim R. R. A. Martins
October 2002

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **OCT 2002**	2. REPORT TYPE	3. DATES COVERED **00-00-2002 to 00-00-2002**

4. TITLE AND SUBTITLE **A Coupled-Adjoint Method for High-Fidelity Aero-Structural Optimization**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Stanford University,Department of Aeronautics and Astronautics,Stanford,CA,94305**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**128**	

To my father,

José Ávila Martins (1917–1996),

this is for you . . .

Abstract

A new integrated aero-structural design method for aerospace vehicles is presented. The approach combines an aero-structural analysis solver, a coupled aero-structural adjoint solver, a geometry engine, and an efficient gradient-based optimization algorithm. The aero-structural solver ensures accurate solutions by using high-fidelity models for the aerodynamics, structures, and coupling procedure. The coupled aero-structural adjoint solver is used to calculate the sensitivities of aerodynamic and structural cost functions with respect to both aerodynamic shape and structural variables. The aero-structural adjoint sensitivities are compared with those given by the complex-step derivative approximation and finite differences. The proposed method is shown to be both accurate and efficient, exhibiting a significant cost advantage when the gradient of a small number of functions with respect to a large number of design variables is needed. The optimization of a supersonic business jet configuration demonstrates the usefulness and importance of computing aero-structural sensitivities using the coupled-adjoint method.

Acknowledgments

I am very fortunate to have had Juan Alonso as my advisor. I will be forever inspired by his contagious enthusiasm and his belief in a potential I didn't know I had. I am also very thankful for having the opportunity to collaborate with Ilan Kroo, my first mentor at Stanford, who introduced to me the fascinating topic of aircraft design optimization. I am also indebted to Professor Antony Jameson — who is really my grand-advisor through two of his former students — for his awe-inspiring achievements in aerodynamic shape optimization. I am thankful for having an unofficial co-advisor in James Reuther, whose energy and sense of humor kept me going at the most critical times. I wish to thank Michael Saunders and Walter Murray for participating in my dissertation committee and David Saunders, for his willingness to help.

During my years at Stanford I benefited from the friendship of my colleagues. With my friend and best man, Peter Sturdza, I rediscovered the thrill of research. To my good friend Nicolas Antoine I owe special thanks for staying up late last night to proofread this dissertation. I am also grateful to my friends Peter Kunz, Gary L. Fay II, Dan Borglund and Martin Chan for making life in and outside the office more fun.

Coming to Stanford would not have been possible without the support of the US-Norway Fulbright Foundation. A significant portion of my doctoral research was funded by the Portuguese government's *Fundação para a Ciência e Tecnologia* and in the later years I benefited from the support of the AFOSR under grant number AF-F49620-01-1-0291.

My father instilled in me the value of higher education. He passed away a few months before I started at Stanford and he would have very much liked to see me finish this work. I am eternally grateful to my mother for having the vision to send me abroad at an early age.

The joy of finishing my thesis is nothing compared to the joy I have experienced since I met my wife Sandra in an incredible stroke of luck six years ago. With Sandra I have found myself, and I look forward to the rest of our lives together.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 MDO in Aircraft Design

A considerable amount of research has been conducted in multidisciplinary design optimization (MDO) and its application to aircraft design. Due to its multidisciplinary nature, aeronautical engineering was one of the first applications of MDO. Aircraft are extremely complex systems whose design requires the detailed consideration of disciplines such as aerodynamics, structures, materials, controls, and propulsion. Wakayama [68], for example, showed that in order to obtain realistic wing planform shapes with aircraft design optimization, it is necessary to include multiple disciplines in conjunction with a complete set of real-world constraints.

Since the inception of MDO, research in this field has yielded many different techniques for inter-disciplinary coupling in both analysis and optimization as well as applications of these techniques to real-world design problems. Surveys by Sobieski and Haftka [65], and Alexandrov and Hussaini [3] present excellent descriptions of the various existing strategies for MDO. A number of ideas for solving complex MDO problems have been developed. These ideas include multilevel optimization strategies [4, 35], collaborative optimization [12, 37, 17], individual discipline feasible methods [16], as well as integrated coupled optimization procedures [23, 49, 43, 45, 44].

Unfortunately, most of the MDO so far has featured low-fidelity models of the participating disciplines. However, during the past decade high-fidelity methods for analysis of complex engineering problems such as those found in fluid dynamics and structural mechanics have reached a mature stage: many difficult numerically intensive problems are now readily solved with modern computer facilities. The aircraft design community is increasingly using computational fluid dynamics (CFD) and computational structural mechanics

(CSM) tools to replace traditional approaches based on simplified theories and wind tunnel testing. With the advancement of these numerical analysis methods well underway, the stage is now set for the integration of these tools with the objective of performing high-fidelity MDO.

One of the defining characteristics of the various MDO strategies is the amount of communication that is required between the disciplines in both the analysis and the optimization procedures. The allowable level of disciplinary autonomy is usually inversely related to the number of variables exchanged in the interdisciplinary coupling. Thus, for problems where large numbers of variables are exchanged between disciplines, it may be necessary to resort to fully integrated MDO, while for problems with low dimensionality coupling, modular strategies may hold an advantage in terms of ease of implementation.

The level of coupling between the disciplines in a multidisciplinary system is problem dependent and significantly affects the choice of MDO strategy. Difficulties also arise from the wide variety of design problems: an approach that is applicable to one problem may not be appropriate for another. High-fidelity aero-structural optimization, for example, features interdisciplinary coupling that requires a large number of design variables. Furthermore, the values of the objective functions and constraints depend on highly coupled multidisciplinary analyses. Therefore, a tightly coupled MDO environment is adopted for the high-fidelity aero-structural optimization problem that constitutes the focus of this work.

1.2 Optimization Methods

Computational design procedures are based on numerical analysis methods that evaluate the relative merit of a set of feasible designs. The merit of a design is based on the value of an *objective function* that is computed using numerical simulations such as CFD and CSM programs. The choice of objective function is extremely important and requires a deep knowledge of the multidisciplinary design problem at hand.

Traditionally, the process of design using numerical simulations has been carried out by trial and error, relying on the intuition and experience of a designer to select variations in the design parameters. However, as the number of design variables is increased, the designer's ability to make the correct choices diminishes. To efficiently span a design space of large dimensionality, numerical simulations needs to be combined with automatic optimization procedures.

A constrained optimization problem can be presented in the general form as,

$$\begin{array}{rccr} \text{minimize} & I(x_j) & & \\ \text{w.r.t} & x_j & j = 1, 2, \ldots, N_x & \\ \text{subject to} & g_m(x_j) \geq 0, & m = 1, 2, \ldots, N_g & \end{array} \quad (1.1)$$

where I is a nonlinear function of N_x design variables x_j, and g_m are the N_g nonlinear inequality constraints we have to satisfy. For a given design problem, a number of parameters, x_j, are allowed to vary when searching for the design with the best figure of merit. Optimization algorithms are concerned with finding the design variables that yield the optimum.

There are a large number of algorithms for numerical optimization, but they all fall into one of two main categories. In the first category we have the *zeroth order methods* which include grid searching, genetic algorithms, neural networks, random searches and simulated annealing. None of these methods rely on any information other than the value of the objective function.

Grid searching is an approach that systematically surveys the design space by evaluating each point in a multidimensional grid. The main problem with this method is that the number of function evaluations required for optimization increases exponentially with the number of design variables and it quickly becomes prohibitively expensive to perform a grid search for more than a few design variables.

In contrast with grid searching, random searches are a non-systematic way of spanning the design space. These methods do not require as many function evaluations, but they do not guarantee that the optimum will be found and are also prone a dramatic increase in cost for large numbers of design variables.

Among gradient-free optimization methods, the nonlinear simplex is one of the most widely used [50]. To create a simplex, $N + 1$ points are evaluated in N-dimensional space. The simplex then moves, expands and contracts as it explores the design space searching for the a better point. This method is very simple and robust, but inefficient for problems with more than half a dozen design variables.

Evolutionary algorithms are another type of optimization methods that offer simplicity and robustness. These algorithms use computational models of evolutionary processes to choose the design parameters. Evolutionary methods are also have the ability to find multiple optimal solutions [71, 52].

In sum, the most significant limitation of zeroth order methods is that, as the number of design parameters increases, the number of function evaluations needed to reach the optimum rapidly increases beyond what is currently computationally feasible.

In a second category, we have *gradient-based methods*, which use not only the value of the objective function but also its gradient with respect to the design parameters. Gradient-based optimization algorithms interprets first and sometimes second order sensitivity information to take steps in the design space that will lead to the optimum. The main advantage of gradient methods is that they converge to the optimum with a significantly smaller number of function evaluations. Unfortunately, these methods only work well when the objective function varies smoothly within the design space and they only guarantee convergence to a local optimum. Steepest descent is the simplest gradient method, where each optimization step is taken in the direction of the gradient vector. Newton methods require second order sensitivity information — the Hessian matrix — in addition to the first derivatives and exhibit a much higher rate of convergence. Quasi-Newton, conjugate gradient, and variable metric strategies approximate the Hessian during the search. Most of these methods use the sensitivity information to identify a search direction in the design space and then perform a one-dimensional search in that direction before finding a new search direction [22, 20, 21].

Both zeroth and first order classes of optimization algorithms have a role in solving engineering problems. In a problem with a limited number of design variables with multiple local minima or discontinuities, it is clear that a zeroth order method is more suitable. On the other hand, many single-discipline high-fidelity aircraft design problems feature a large number of design variables and a smooth design space. These problems are amenable to the use of gradient-based optimization algorithms. In particular, gradient methods are used extensively for aerodynamic shape optimization problems because these are often parameterized with hundreds of design variables and require computationally expensive high-fidelity analyses. With a few notable exceptions [62, 52], such requirements make the use of zeroth order methods infeasible for high-fidelity aerodynamic shape optimization problems.

In this dissertation, a gradient-based strategy is employed, enabling the use of hundreds or even thousands of design parameters to achieve near optimal shape-structure combinations. This level of design detail — which is arguably necessary at the end of the preliminary design stage — cannot be treated by non-gradient methods as the analyses involve the solution of a high-fidelity aero-structural system. The optimization algorithm used herein is NPSOL [20], a state-of-the-art solver for nonlinear programming which is well-suited for our problem.

1.3 Sensitivity Analysis

Sensitivity analysis consists in computing derivatives of one or more quantities (outputs) with respect to a number of independent variables (inputs). Although there are various uses for sensitivity information, our main motivation is the use of this information in gradient-based optimization. Since the calculation of gradients is often the most costly step in the optimization cycle, using efficient methods that accurately calculate sensitivities is extremely important for timely design methods.

There are a several different methods for sensitivity analysis but since none of them is the clear choice for all cases, it is important to understand their relative merits. When choosing a method for computing sensitivities, one is mainly concerned with its accuracy and computational expense and scalability for increasing numbers of independent variables. Factors that affect the choice of method include: the ratio of the number of outputs to the number of inputs, the importance of computational efficiency and the amount of human effort that is required in the implementation.

1.3.1 Finite Differences

Finite-difference formulae are commonly used to estimate sensitivities. Although these approximations are neither particularly accurate nor computationally efficient, the greatest advantage of this method resides in the ease of implementation.

All finite-difference approximations can be derived by truncating a Taylor series expanded about a given point x_j. A common estimate for the first derivative is the *forward difference* which is given by

$$\frac{\mathrm{d}f(x_j)}{\mathrm{d}x_j} = \frac{f(x_j + h) - f(x_j)}{h} + \mathcal{O}(h), \tag{1.2}$$

where h is the finite-difference interval, or step. The truncation error is $\mathcal{O}(h)$, and hence this is a first-order approximation. Higher-order approximations — such a the second-order accurate central difference — can be obtained by combining different Taylor series expansions, but more accurate approximations require more function evaluations.

When estimating sensitivities using finite-difference formulae we are faced with the “step-size dilemma”, i.e. the desire to choose a small step size to minimize truncation error while avoiding the use of a step so small that errors due to subtractive cancellation become dominant.

The cost of calculating sensitivities with finite-differences is proportional to the number of design variables since f must be calculated for each perturbation of x_j. This means that if we use forward differences, for example, the cost would be $N_x + 1$ times the cost of calculating f. For central differences, the cost would be almost doubled, i.e., $2N_x$.

1.3.2 Complex-Step Derivative Approximation

The complex-step derivative approximation is a relatively new method that unlike finite differences is extremely robust to changes in the step size [46, 47]. The first derivative approximation can be derived from complex calculus and is represented by the simple formula

$$\frac{\mathrm{d}f(x_j)}{\mathrm{d}x_j} = \frac{\mathrm{Im}[f(x_j + ih)]}{h} + \mathcal{O}(h^2), \tag{1.3}$$

where we take the imaginary part of the function evaluation of the design perturbed by a pure imaginary step and divide it by h to obtain a second order approximation. The complex-step method is the subject of Chapter 3. As in the case of finite-differencing, the cost for evaluating a gradient is proportional to the number of variables, N_x. However, since complex arithmetic is required, this cost is usually more than twice the cost of forward differencing.

1.3.3 Algorithmic Differentiation

Algorithmic differentiation — also known as computational differentiation or automatic differentiation — is a well-established method based on the systematic application of the chain rule of differentiation to each operation in the program flow [25, 15]. For each intermediate variable in the algorithm, a variation due to one input variable is carried through, and all the required sensitivities are computed in one program call. The derivatives given by the chain rule can be propagated forward (*forward mode*) or backwards (*reverse mode*).

The forward mode of algorithmic differentiation is shown to be very similar to the way the complex-step method works and the cost of computing the gradient is of the same order, i.e., proportional to the number of design variables. These similarities are discussed at length in Chapter 3.

In reverse mode, algorithmic differentiation executes the code forward and then backward to calculate derivatives of one function with respect to N_x design variables. The total number of operations is independent of N_x, but the memory requirements may be prohibitive, especially for the case of large iterative algorithms.

1.3.4 Analytic Methods

Analytic approaches such as adjoint and direct methods are the most accurate and efficient for sensitivity analysis. They are, however, more involved than the other approaches presented so far because they require knowledge of the governing equations and the algorithm that is used to solve them, in order to derive and implement a program that solves the corresponding sensitivity equations.

Adjoint methods are particularly attractive since the cost of computing the gradient of a given function is independent of the number of design variables. Due to the large numbers of variables needed for high-fidelity parameterization, an adjoint method for the aero-structural system is the obvious choice for use in optimization. The coupled-adjoint method for the aero-structural governing equations is developed in Chapter 4.

1.4 High-Fidelity Aero-Structural Optimization

1.4.1 Previous Work on Aerodynamic Shape Optimization

During the late 1980's and early 90's, the maturation of computational fluid dynamics (CFD) tools and the ever-increasing capability of computer systems enabled the use of CFD in design [32]. An adjoint method was first applied to the governing equations for transonic flow by Jameson [30]. This breakthrough enabled gradient-based aerodynamic shape optimization with respect to many more design variables than was previously feasible. Since then, high-fidelity aerodynamic shape optimization has been extremely successful, yielding optimized designs of airfoils [31], wings [60, 33], and full aircraft configurations [59, 55, 56].

The performance of transonic and supersonic designs can be extremely sensitive to subtle changes in the aerodynamic shape. In the case of a transonic configuration, such as the business jet shown in Figure 1-1, there is a strong shock on the top surface of the baseline wing. It is possible to smooth this shock and drastically reduce the wave drag of the configuration by applying the right combination of small shape variations. Figure 1-2 shows the result of such variations applied to an airfoil by using an adjoint method in conjunction with a gradient-based optimizer.

In the case of a supersonic configuration, such as the high-speed civil transport (HSCT) shown in Figure 1-3, high-fidelity methods are necessary to analyze full configurations that account for the effect of the nacelles on the design. To take advantage of favorable interference between the nacelles and the wing, subtle variations in the wing and nacelle shapes as well as the position of the nacelles are necessary. Figure 1-3 shows both the baseline and

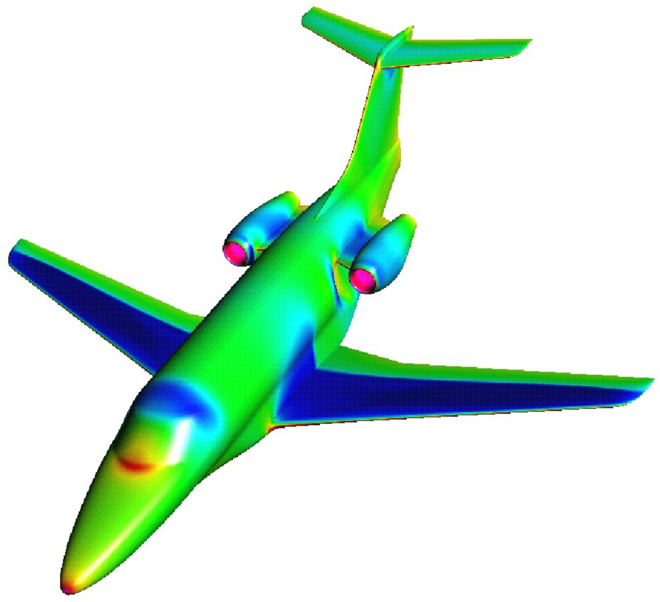

Figure 1-1: Surface pressure on a transonic business jet configuration.

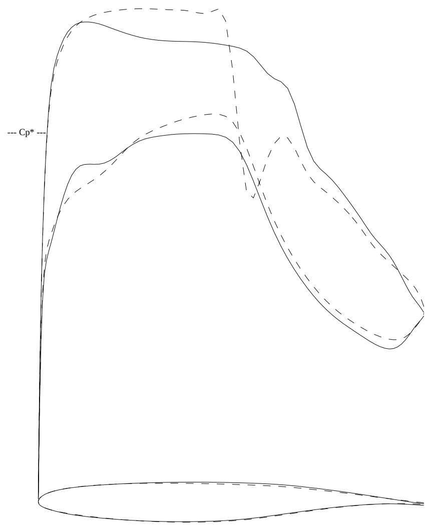

Figure 1-2: Baseline and optimized airfoil and pressure coefficient distributions. Baseline represented by dashed lines.

optimized designs, where the most noticeable improvement is the weakening of the nacelle shocks.

When performing high-fidelity aerodynamic shape optimization of both transonic and supersonic aircraft configurations, a large number of design parameters is required to take full advantage of this design methodology.

Since aircraft are complex multidisciplinary systems, aerodynamic shape optimization alone does not suffice for aircraft design. When defining an aerodynamic shape optimization problem, geometric constraints are often used to prevent impractical designs. In supersonic wing design, for example, if minimum airfoil thickness constraints are not imposed, aerodynamic optimization will decrease the thickness-to-chord ratio of the wing sections to the extend that it becomes impossible to design a structure that would fit inside the optimized wing and be able to sustain the required loads.

1.4.2 Motivation for Integrated Aero-Structural Optimization

Within this dissertation, the main motivation for integrating structural design capability with the existing aerodynamic shape optimization framework was to eliminate the need for the artificial geometric constraints that are imposed when performing aerodynamic optimization alone.

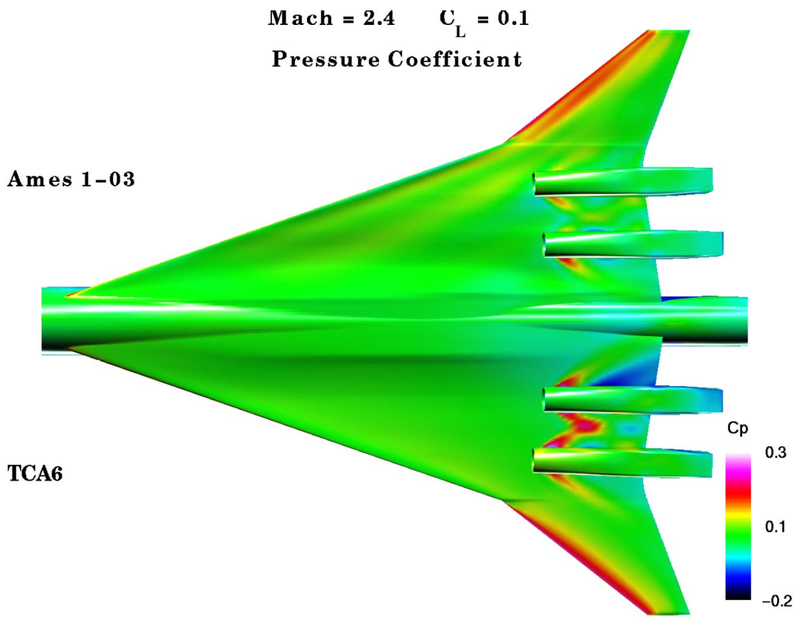

Figure 1-3: Surface pressure coefficient of baseline (bottom) and optimized (top) configurations of a supersonic civil transport.

Aero-structural design has traditionally been performed in a cut-and-try basis: aerodynamicists have an idea of the shape of an “optimal” load distribution and then tailor the jig shape of the structure so that the deflected wing shape under a 1-g load gives the desired load distribution. This approach may suffice for conventional transport aircraft, where there is considerable accumulated experience. In the case of either new planform concepts or new flight regimes, however, the lack of experience combined with the complexities of aero-structural interactions can lead to designs that are far from optimal.

In the worst case, an ill-informed designer might try to perform sequential optimization of the participating disciplines. This sequential approach is illustrated in Figure 1-4 for the case of aerodynamics and structures. Since the aerodynamic optimization is not aware of the effect that its design variables have on the structural weight, it always yields an elliptic lift distribution to minimize the drag. The structural optimization, on the other hand, finds the minimum structural weight for fixed aerodynamic variables. This sequential optimization process usually converges, but the final result is not the true optimum of the coupled system.

For maximum lift-to-drag ratio, it is a well-known result from classical aerodynamics that a wing must exhibit an elliptic lift distribution. For aircraft design, however, it is usually not the lift-to-drag ratio we want to maximize but an objective function that reflects

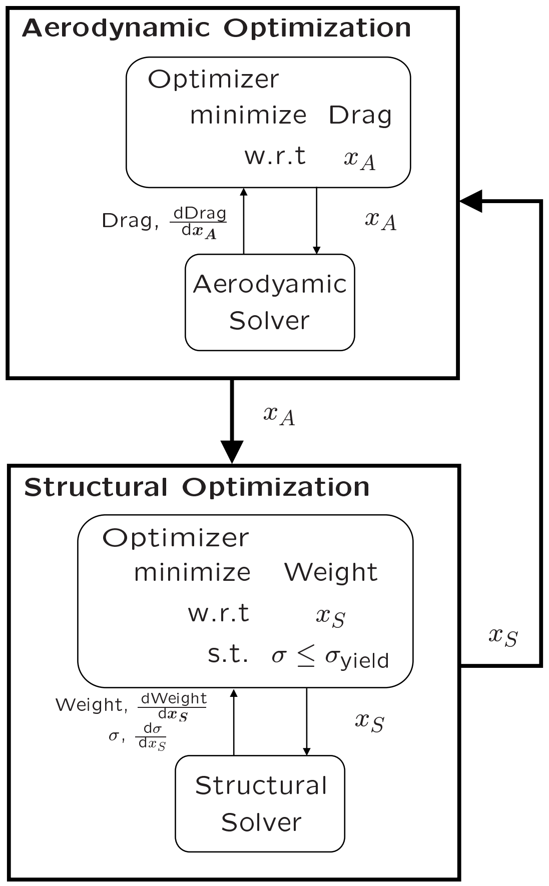
Aerodynamic Optimization
Optimizer
minimize Drag
w.r.t x_A
Drag, $\frac{\text{dDrag}}{\text{d}x_A}$
x_A
Aerodyamic Solver
x_A
Structural Optimization
Optimizer
minimize Weight
w.r.t x_S
s.t. $\sigma \leq \sigma_{\text{yield}}$
Weight, $\frac{\text{dWeight}}{\text{d}x_S}$
σ, $\frac{\text{d}\sigma}{\text{d}x_S}$
x_S
Structural Solver
x_S

Figure 1-4: Diagram representing sequential optimization of aerodynamics and structures.

the overall mission of the particular aircraft. Consider, for example, the Breguet range formula for jet-powered aircraft

$$\text{Range} = \frac{V}{c}\frac{C_L}{C_D}\ln\frac{W_i}{W_f}, \tag{1.4}$$

where V is the cruise velocity and c is the thrust specific fuel consumption of the powerplant. C_L/C_D is the ratio of lift to drag, and W_i/W_f is the ratio of initial and final cruise weights of the aircraft.

The Breguet range equation expresses a trade-off between the drag and the empty weight of the aircraft and constitutes a reasonable objective function to use in aircraft design. To perform aero-structural optimization we can parameterize the aircraft configuration with both aerodynamic and structural variables and then maximize the range for a fixed initial

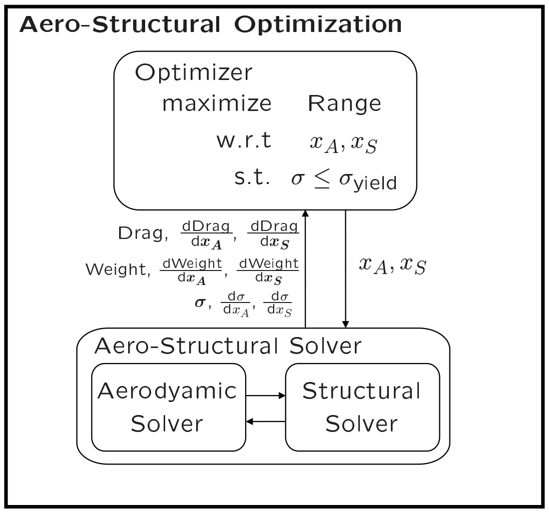
Aero-Structural Optimization
Optimizer
maximize Range
w.r.t x_A, x_S
s.t. $\sigma \leq \sigma_{\text{yield}}$
Drag, $\frac{\text{dDrag}}{\text{d}x_A}$, $\frac{\text{dDrag}}{\text{d}x_S}$
Weight, $\frac{\text{dWeight}}{\text{d}x_A}$, $\frac{\text{dWeight}}{\text{d}x_S}$
σ, $\frac{\text{d}\sigma}{\text{d}x_A}$, $\frac{\text{d}\sigma}{\text{d}x_S}$
x_A, x_S
Aero-Structural Solver
Aerodyamic Solver
Structural Solver

Figure 1-5: An integrated approach to aero-structural optimization.

cruise weight, subject to stress constraints by solving a single optimization. A representation of this integrated, all-at-once approach is represented in Figure 1-5.

Solving this type of aero-structural optimization problem using an integrated approach, we obtain the spanwise lift distribution shown in Figure 1-6. This particular optimization problem employed a panel code for aerodynamics and a one-spar model for the structure of the wing. The details of this simple aero-structural optimization problem and additional results are described in Appendix A.

This optimum lift distribution when considering both the aerodynamics and the structures trades off the drag penalty associated with unloading the tip of the wing, where the loading contributes most to the maximum stress at the root of the wing structure, in order to reduce the weight. The end result is an increase in range when compared to the elliptically loaded wing that results from an increased weight fraction W_i/W_f. The result shown in Figure 1-6 illustrates the need for taking into account the coupling of aerodynamics and structures when performing aircraft design.

Despite significant accomplishments in single-discipline optimization in the field of aircraft design [55, 56, 2], progress towards the development of high-fidelity MDO methods has been slow, and the modeling of the participating disciplines in most of the MDO applications that have appeared so far has remained at a relatively low level. While useful at the conceptual design stage, lower-order models cannot accurately represent a variety of nonlinear phenomena such as shocks on transonic wings and shock interference in complex

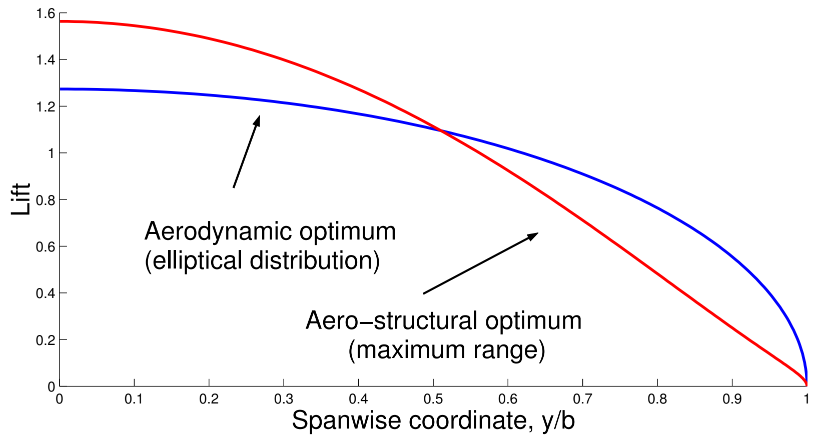

Figure 1-6: Elliptic vs. aero-structural optimum lift distribution.

supersonic configurations, which can play an important role in the search for the optimum design.

This is certainly the case in the design of both small and large supersonic transports, where simple beam theory models of the wing cannot be used to accurately describe the behavior of the wing structure. In some cases, these aircraft cruise significant portions of their flight at different Mach numbers. In addition, a variety of studies show that supersonic transports exhibit a range of undesirable aeroelastic phenomena due to the low bending and torsional stiffness that result from wings with low thickness to chord ratio. These phenomena can only be suppressed when aero-structural interactions are taken into account at the preliminary design stage [8].

An exception to the use of low-fidelity modeling in MDO is the recent work by Giunta [23] and by Maute et al. [48, 49] where aero-structural sensitivities are calculated using higher-fidelity models.

1.4.3 Proposed Aircraft Optimization Problem

The objective of this work is to develop techniques for efficiently computing multidisciplinary sensitivities and use these sensitivities to enable high-fidelity aero-structural optimization of aircraft configurations. The optimization problem proposed in this section is very simplistic from the point of view of aircraft design, since it does not take into account engine sizing, trajectory, tail design, and other disciplines that are part of the complete aircraft design optimization problem. This problem is intended to be a proof-of-concept design case that demonstrates the viability of high-fidelity aircraft design. Since two of the most computationally intensive disciplines are taken into account, adding the remaining disciplines to form a framework for aircraft design would be easy by comparison.

Design Parameterization

The aircraft configuration is parameterized using two types of design variables. The first type of variable modifies the outer mold line (OML) of the configuration while the second type of variable controls the sizing of underlying structure.

The OML design variables can be applied to any of the components used to define the aircraft geometry. For each wing-like component (main wing, canard, horizontal tail, etc.), the shape is modified at a number of specified airfoil sections. Each of these sections may be independently modified while the spanwise resolution can be controlled by the number and position of the sections. The shape modifications to the airfoils are linearly lofted between stations. Various types of design variables may be applied to the airfoils: twist, leading and trailing edge droop, and Hicks–Henne bump functions, among others. The Hicks–Henne functions are of the form

$$b(\zeta) = x_n \left[\sin\left(\pi \zeta^{\frac{\log \frac{1}{2}}{\log t_1}} \right) \right]^{t_2}, \tag{1.5}$$

where t_1 is the location of the maximum of the bump in the range $0 \leq \zeta \leq 1$ at $\zeta = t_1$, since the maximum occurs when $\zeta^\alpha = 1/2$, where $\alpha = \log(1/2)/\log t_1$. The parameter t_2 controls the width of the bump. The advantage of these functions is that when they are applied to a smooth airfoil, that airfoil remains smooth.

The structural design variables consists of the thicknesses of the structural finite elements. The topology of the structure remains unchanged, i.e., the number of spars and ribs and their position are fixed throughout the optimization. Using any of these discrete parameters as design variables results in a discontinuous design space that is not compatible with the gradient-based design approach. However, because the OML determines the location of the nodes of the structural model, variations of the OML have an effect on the depth of the spars and ribs of the wing box.

Objective Function

From the detailed mission analysis of a particular aircraft design it is usually possible to find the correct trade-off between aerodynamic drag and structural weight. This means that we can optimize a design by minimizing the objective function,

$$I = \alpha C_D + \beta W, \tag{1.6}$$

where C_D is the drag coefficient, W is the structural weight and α and β are scalar parameters.

To perform gradient-based optimization, we need the sensitivities of the objective function (1.6) with respect to all the design variables. Since this objective function is a linear combination of the drag coefficient and the structural weight, its sensitivity can be written as

$$\frac{\mathrm{d}I}{\mathrm{d}x_n} = \alpha \frac{\mathrm{d}C_D}{\mathrm{d}x_n} + \beta \frac{\mathrm{d}W}{\mathrm{d}x_n}. \tag{1.7}$$

The sensitivity of the structural weight, $\mathrm{d}W/\,\mathrm{d}x_n$ is trivial, since the weight calculation is independent of the aero-structural solution. This gradient is calculated analytically for the structural thickness variables and by finite differences for the OML variables. The drag coefficient sensitivity, $\mathrm{d}C_D/\,\mathrm{d}x_n$, is not this trivial since it does depend on the aero-structural solution.

Constraints

Among the imposed constraints, the most obvious is that lift must always equal the weight of the aircraft during cruise. In our optimization problem we constrain the C_L by periodically adjusting the angle-of-attack of the aircraft within the aero-structural solution until the desired lift is obtained within a specified tolerance. Otherwise, OML design changes would quickly result in lower drag coefficients simply because of reduced lift. Some design problems require that the objective function be minimized over a range of operating conditions (multipoint design). In these situations, the appropriate lift constraint would be imposed at each design point using the same procedure.

In addition to maintaining the C_L, the stresses are also constrained so that the yield stress of the material is not exceeded at various load conditions. There are typically thousands of finite elements describing the structure of the aircraft, and it becomes computationally very costly to treat these constraints separately. The difficulty of the problem is that even though there are efficient ways of computing sensitivities of a few functions with respect to many design variables, and of computing sensitivities of many functions with respect to a few design variables, there is no known efficient method for computing sensitivities of many functions with respect to many design variables. Thus, we are left to choose between treating a large number of design variables or being able to handle multiple cost functions and constraints.

For this reason, we lump the individual element stresses using Kreisselmeier–Steinhauser (KS) functions. In the limit, all element stress constraints can be lumped into a single KS function, thus minimizing the cost of a large-scale aero-structural design cycle. Suppose that we have the following constraint for each structural finite element,

$$g_m = 1 - \frac{\sigma_m}{\sigma_y} \geq 0, \tag{1.8}$$

where σ_m is the element von Mises stress and σ_y is the yield stress of the material. The corresponding KS function is defined as

$$\mathrm{KS}\left(g_m\right) = -\frac{1}{\rho} \ln \left(\sum_m \mathrm{e}^{-\rho g_m} \right). \tag{1.9}$$

This function represents a lower bound envelope of all the constraint inequalities and ρ is a positive parameter that expresses how close this bound is to the actual minimum of the constraints. This constraint lumping method is conservative and may not achieve the same optimum that a problem treating the constraints separately would. However, the practicality of KS functions has been demonstrated and constitutes a viable alternative, being effective in optimization problems with thousands of constraints [64, 2].

Problem Statement

Having defined our objective function, design variables and constraints, we can now summarize the aircraft design optimization problem as follows:

$$\begin{aligned} \underset{x_n \in \mathbb{R}^n}{\text{minimize}} \quad & I = \alpha C_D + \beta W \\ \text{subject to} \quad & C_L = C_{L_T} \\ & \mathrm{KS} \geq 0 \\ & x_n \geq x_{n_{\min}}. \end{aligned} \tag{1.10}$$

The stress constraints in the form of KS functions must be enforced by the optimizer for aerodynamic loads corresponding to a number of flight and dynamic load conditions. Finally, a minimum gage is specified for each structural element thickness.

Note that it is possible to generalize this problem to account for multiple flight conditions by including the weighted sum of all the drag coefficients in the objective function. The weights would determine the relative importance of each design point. It is also possible to consider multiple load conditions by enforcing a KS constraint for each load case.

1.5 Dissertation Layout

The rest of this dissertation starts with the description of the aero-structural solver and its components in Chapter 2. Chapter 3 is dedicated to the complex-step derivative approximation, which is later used for providing benchmark sensitivities in the validation of the coupled-adjoint method. Results comparing the accuracy and efficiency of the complex-step method with finite-differences and algorithmic differentiation are also presented in this chapter. A discussion of analytic methods for sensitivity analysis of multidisciplinary systems is presented in Chapter 4, along with a discussion of the implementation of the coupled-adjoint approach for the aero-structural design framework is discussed. The validation and evaluation of the results given by the aero-structural adjoint approach are also presented. Then, in Chapter 5, the aero-structural optimization of a supersonic business jet configuration is performed using the gradients computed by the coupled-adjoint method. Finally, the conclusions of this work are made in Chapter 6.

1.6 Contributions

The main contributions of this dissertation lie on the theory, implementation and practical application of two different sensitivity analysis methods: the complex-step derivative approximation and the coupled-adjoint method.

The theory behind the complex-step derivative approximation is thoroughly explored and new insights are presented. A methodology for automating the implementation this method is developed and is used to compute sensitivities of various solvers, including the aero-structural solver to which the coupled-adjoint method is applied.

The theory for analytic sensitivity analysis for general multidisciplinary systems is unified by deriving four different methods starting from the same basic equations, including a previously unpublished method: the alternate coupled-adjoint method. By adopting this unified approach to deriving the multidisciplinary sensitivity equations, we are able to gain a broader understanding of this topic and to discuss the relative efficiency and practicality of the various methods. One of these — the coupled-adjoint method — is implemented in a framework for aero-structural optimization of aircraft configurations, and practical results pertaining to a supersonic business jet configuration are presented.

Chapter 2

Aero-Structural Analysis

This chapter describes the analysis tools that are used in the aero-structural design framework: the flow solver, the structural solver, and the procedure for interdisciplinary coupling. In each of these components, high-fidelity modeling is a requirement to maintain a high level of overall accuracy.

2.1 Computational Fluid Dynamics

The aerodynamic analysis and design module, SYN107-MB [55], is a multiblock parallel flow solver that is applicable to both the Euler and the Reynolds Averaged Navier–Stokes equations. This solver represents the state-of-the art, being accurate and efficient for the computation of the flow around full aircraft configurations [58, 70]. This module also includes the mesh perturbation algorithm that is detailed in Section 2.3.2.

Since SYN107-MB is a package for aerodynamic shape optimization, it also includes an adjoint solver for aerodynamic sensitivity analysis. This capability is out of the scope of the present chapter and therefore is described in Chapter 4, which focuses on the sensitivity analysis.

In this work, the flow solver is used exclusively in Euler mode. The flow domain is discretized using a multiblock structured grid with hexahedral cells and the flow state variables are evaluated at the center of these cells. This cell-centered discretization is used in conjunction with a finite-volume scheme to solve the flow equations numerically.

The vector of flow state variables at a given cell center is defined as,

$$\boldsymbol{W} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{bmatrix}, \tag{2.1}$$

where ρ is the flow density, u_1, u_2 and u_3 are the three Cartesian components of velocity, and E is the total (internal plus kinetic) energy. The flux vector is defined as,

$$\boldsymbol{F}_l = \begin{bmatrix} \rho u \\ \rho u_1 u_l + p\delta_{1l} \\ \rho u_2 u_l + p\delta_{2l} \\ \rho u_3 u_l + p\delta_{3l} \\ \rho H u_l \end{bmatrix}, \tag{2.2}$$

for the three spatial directions $l = 1, 2, 3$. The pressure p is related to the total energy E by the equation of state for an ideal gas,

$$p = (\gamma - 1)\rho \left(E - \frac{\boldsymbol{u}^2}{2} \right), \tag{2.3}$$

where γ is defined as the ratio of specific heats. The total enthalpy is also related to the total energy and pressure by,

$$H = E + \frac{p}{\rho} = \frac{c^2}{\gamma - 1} + \frac{\boldsymbol{u}^2}{2}, \tag{2.4}$$

where c is the isentropic speed of sound given by,

$$c^2 = \frac{\gamma p}{\rho}. \tag{2.5}$$

The semi-discrete form of the flow governing equations can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}(V_{ijk}\boldsymbol{W}_{ijk}) + \boldsymbol{R}_{ijk} = 0, \tag{2.6}$$

where V_{ijk} is the volume of the cell, $\boldsymbol{W}_{ijk}$ is the state vector, and $\boldsymbol{R}_{ijk}$ is the net flux out of the cell. The indices i, j, k denote the position of the cell in a block of the three-dimensional mesh.

For the cell-centered scheme this flux is given as,

$$\begin{aligned} \boldsymbol{R}_{ijk} = {} & \boldsymbol{F}_{i+1/2,j,k} \cdot \boldsymbol{S}_{i+1/2,j,k} - \boldsymbol{F}_{i-1/2,j,k} \cdot \boldsymbol{S}_{i-1/2,j,k} \\ & + \underset{i,j+1/2,k}{\boldsymbol{F}} \cdot \boldsymbol{S}_{i,j+1/2,k} - \boldsymbol{F}_{i,j-1/2,k} \cdot \boldsymbol{S}_{i,j-1/2,k} \\ & + \boldsymbol{F}_{i,j,k+1/2} \cdot \boldsymbol{S}_{i,j,k+1/2} - \boldsymbol{F}_{i,j,k-1/2} \cdot \boldsymbol{S}_{i,j,k-1/2}, \end{aligned} \tag{2.7}$$

where $\boldsymbol{F}_{i+1/2,j,k}$ is the flux vector on the surface, and $\boldsymbol{S}_{i+1/2,j,k}$ is the vector normal to the face where the flux is being evaluated and its magnitude is equal to the area of the face. The flux vector on each face is evaluated by averaging the values at the center of the cells on either side of the surface, i.e.,

$$\boldsymbol{F}_{i+1/2,j,k} = \frac{1}{2}(\boldsymbol{F}_{i,j,k} + \boldsymbol{F}_{i+1,j,k}). \tag{2.8}$$

This central-difference approximation ensures that for sufficiently smooth meshes, this scheme is second-order accurate.

Since the finite-volume scheme is vulnerable to numerical instabilities, numerical dissipation is necessary to damp any spurious oscillations. Numerical dissipation is implemented by adding an additional term to the semi-discrete form of the flow governing equations (2.6), i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}t}(V_{ijk}\boldsymbol{W}_{ijk}) + \boldsymbol{R}_{ijk} - \boldsymbol{D}_{ijk} = 0, \tag{2.9}$$

where, $\boldsymbol{D}_{ijk}$ represents all the dissipative terms across the faces of a cell, and can be written as,

$$\begin{aligned} \boldsymbol{D}_{ijk} = {} & \boldsymbol{d}_{i+1/2,j,k} - \boldsymbol{d}_{i-1/2,j,k} + \boldsymbol{d}_{i,j+1/2,k} \\ & - \boldsymbol{d}_{i,j-1/2,k} + \boldsymbol{d}_{i,j,k+1/2} - \boldsymbol{d}_{i,j,k-1/2}. \end{aligned} \tag{2.10}$$

The details of the term $\boldsymbol{d}_{i+1/2,j,k}$ depend on the particular artificial dissipation scheme. In this case, the Jameson–Schmidt–Turkel (JST) scheme [34] is used. This technique for artificial dissipation combines monotonicity and higher order accuracy by blending low and high order dissipative terms using a pressure sensor. The JST scheme, with small modifications, can be made to conform to the local extrema diminishing (LED) principle, whereby local maxima of the solution are not allowed to increase and local minima not allowed to decrease.

To increase the computational efficiency of the flow solver, the multigrid scheme is used.

This scheme is used to accelerate the convergence of iterative algorithms by using corrections obtained from the solutions in a succession of grids that are coarser than the original one. A larger time step can be used in the coarser grids resulting not only in the acceleration of the convergence, but also in more efficient damping of low frequency error modes. The adjoint solver described in the next chapter also uses a multigrid scheme.

2.2 Computational Structural Mechanics

To analyze the structure of the aircraft wing, we use FESMEH, a finite element solver developed by Holden [28]. The package is a linear finite-element solver that incorporates two element types and computes the structural displacements and stresses of wing structures. Although this solver is not as general as some commercially-available packages, it is still representative of the challenges involved in using large models with tens of thousands of degrees of freedom.

The main reason for choosing this package was the availability of the source code, which was a requirement for developing the analytic structural sensitivity analysis methods described in Chapter 4.

2.2.1 Finite-Element Method

The finite-element method is a technique that discretizes the structure into one of more sets of basic structural components Each set exhibits a similar geometry and physical assumption that corresponds to a specific type of finite element. There is a large number of types of elements available and the choice depends on the relative importance of accuracy and cost of the analysis.

The finite elements are connected to adjacent elements by nodal points. Acting at each nodal point are nodal forces and the node is subjected to displacements which represent the degrees-of-freedom. Physically assembling these elements to form the whole structure is equivalent to superimposing these element equations mathematically, The result is a large set of simultaneous equations which can be solved numerically. After applying the loads and boundary conditions, the set of equations can be written as

$$\boldsymbol{K}\boldsymbol{u} - \boldsymbol{f} = 0. \tag{2.11}$$

Here, $\boldsymbol{K}$ is the global stiffness matrix of the structure, $\boldsymbol{u}$ is the vector of nodal displacements, and $\boldsymbol{f}$ is the vector of applied nodal forces. The stiffness matrix is assembled by superposing the element stiffness matrices and it is symmetric and non-singular provided that the rows and columns corresponding to fixed degrees of freedom are eliminated.

For problems with a relatively small number of degrees-of-freedom, i.e. $\mathcal{O}(10^2)$, a Cholesky factorization is appropriate to solve for the unknown displacements. In this case the matrix factorization can be stored explicitly and used multiple times with different load vectors during an aero-structural calculation, assuming that the structure exhibits a linear behavior and that it remains unchanged. For models with large numbers of degrees-of-freedom, a sparse Cholesky factorization or sparse LU decomposition is more appropriate.

Substituting the nodal displacements back into each element formulation provides the distributions of stress and displacements within each element. To determine the displacement field within an element, finite-element theory assumes shape functions that interpolate the nodal displacements, the simplest of which are linear functions. The displacement of any point $\boldsymbol{x}$ in the finite element can be written as

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{\eta}(\boldsymbol{x})\bar{\boldsymbol{u}} \tag{2.12}$$

where $\boldsymbol{\eta}(\boldsymbol{x})$ represents the finite-element shape functions and $\bar{\boldsymbol{u}}$ is the vector of nodal displacements for that element.

There is also a need to calculate the stresses in the structure, which are a function of the displacements. In order to derive the relationship between stresses and displacements, we start with the stress-strain relationship for a given element,

$$\boldsymbol{\sigma} = \boldsymbol{E}\boldsymbol{\varepsilon}. \tag{2.13}$$

This equation is — in the general case — a matrix equation, depending on the dimensionality of the element stresses and strains. The matrix $\boldsymbol{E}$ is usually referred to as the constitutive law matrix.

The strains at the element level depend on the nodal displacements of the finite element, i.e.,

$$\boldsymbol{\varepsilon} = \boldsymbol{b}\bar{\boldsymbol{u}}, \tag{2.14}$$

where $\bar{\boldsymbol{u}}$ denotes the vector of displacements in the local element coordinates, i.e., the coordinate system aligned with the element. The matrix $\boldsymbol{b}$ for a given type of finite element is obtained by differentiating the shape functions of that element.

The displacements in local coordinates are obtained from the displacements in the global coordinates using the coordinate transformation,

$$\bar{\boldsymbol{u}} = \boldsymbol{\lambda u}. \tag{2.15}$$

From equations(2.13 – 2.15) the stresses in each element are then related to the displacements by the following matrix equation,

$$\boldsymbol{\sigma} = \boldsymbol{S u}, \tag{2.16}$$

where S represents the product of the constitutive law matrix, the nodal displacement-strain matrix and the local-to-global coordinate transformation matrix, i.e.,

$$\boldsymbol{S} = \boldsymbol{E b \lambda}. \tag{2.17}$$

2.2.2 Wing Structural Model

This solver models a wing with multiple spars, shear webs, and ribs located at various spanwise stations, and the skins of the upper and lower surfaces of the wing box.

In the modeling of a typical wing structure, triangular plates are used to model the wing skins. Plates are also used for the shear webs of spars and ribs, while the upper and lower spar caps are modeled using truss elements. The wing model in our case consisted of 6 spars and 10 ribs, adding up to a total of 132 nodes, 396 degrees-of-freedom and 640 elements. Figure 2-1 shows the geometry of the finite element structural model.

Two types of finite elements are used: truss and triangular plane-stress plate elements. Both element types have 3 translational degrees of freedom per node, so the truss has a total of 6 degrees of freedom and the plate has 9 degrees of freedom. The main limitation when using these types of elements is that each of the nodes must be simply supported, implying that we can have only one set of plate elements between any two spars.

In this structural model, the body forces due to the mass of the wing and the fuel inside the wing are ignored. If taken into account, these forces would counteract the loads due to lift and reduce the bending moment and shear forces in the structure.

2.3 Aero-Structural Coupling

The coupling of the flow and structural solvers is one of most important components of the aero-structural design framework. Since the disciplinary solvers are of high fidelity, it is important to maintain the same level of accuracy in the coupling.

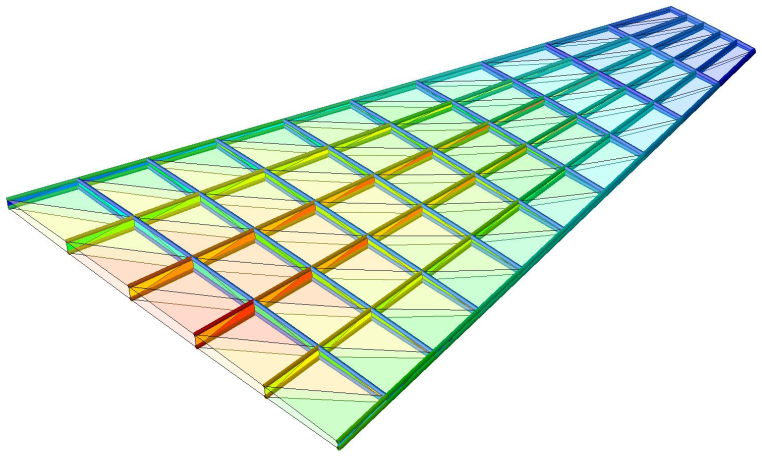

Figure 2-1: Structural model of the wing. Note that the skins are transparent, revealing the spars and ribs inside the structural box.

Two issues in the transfer of information between disciplines are of utmost importance to the success of an automatic design technique. The first one is that the level of fidelity in the coupling of both disciplines has to be considered carefully in order to guarantee that the accuracy of the individual disciplines is not jeopardized. The second one is that the discretization in each discipline — the flow solver mesh and the finite-element model in this case — must preserve geometric consistency during the design process.

A simple diagram depicting the coupling between the two solvers is shown in Figure 2-2. There is a mutual dependence in the state of the aero-structural system: the flow solution is affected by the position and shape of the solid boundary which is dictated by the structural displacements. These displacements in turn depend on the forces applied to the structure due the flow pressures at that boundary. An intermediate module, the *geometry engine* is also shown in the figure. This module is responsible for keeping a centralized *geometry database* of the current geometry as well as for the transfer of the loads and displacements between disciplines. We now describe in detail each of these three modules.

2.3.1 Geometry Engine and Database

The aircraft is surrounded by fluid which is separated from the structure by the fluid-structure interface. Therefore, there is a well-defined surface in three-dimensional space which constitutes the outer-mold line (OML). Because of the importance of the OML in

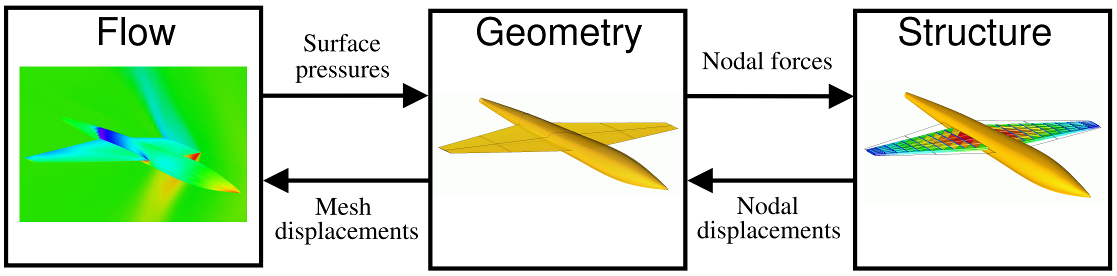

Figure 2-2: Simplified view of the aero-structural coupling.

aero-structural analysis and design problems, a separate utility — *Aerosurf* — is used to generate and manage the OML. *Aerosurf* was specifically created for the analysis and design of aircraft configurations [55, 56, 57].

The baseline geometry of an aircraft configuration is given to *Aerosurf* in the form of separate components, each one being described by a series of cross-sections in three-dimensional space. These components can be fuselages, pylons, nacelles, and wing-like surfaces. Figure 2-3 shows these components for the full configuration of a transonic business jet. Wing, fuselage, empennage, pylons and nacelles are the components in this case.

After lofting the sections that define each component using a bi-cubic spline method, *Aerosurf* intersects these components and divides the resulting surface into a series of patches. At this stage, *Aerosurf* creates a parametric description of each patch and then distributes points on their surface, forming a fine structured watertight mesh. Thus, the set of points formed by the grids of all patches represents a discretization of the OML within *Aerosurf*. The intersected geometry for the business jet is shown in Figure 2-4.

In addition to providing a high-fidelity description of the aircraft geometry, *Aerosurf* also manages a centralized database for the analysis and design of the aircraft. During analysis, any information that needs to be exchanged through the fluid-structure interface — such as pressures or structural displacements — is interpolated onto the OML points. Changes in the OML shape can be due to either structural displacements during aero-structural analysis or changes in shape design variables between design cycles. While the changes in the OML that are due to structural displacements are transferred directly to the OML points, changes due to shape design variables are applied to the un-intersected components first and then these components are re-intersected, creating a new discretized representation of the OML.

For the specific case of aero-structural coupling, the interaction occurs over the OML. Since the OML is defined to have no gaps, the aerodynamic pressure and the structural

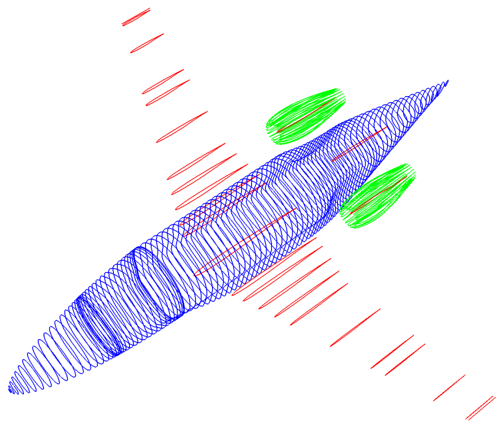

Figure 2-3: Un-intersected components of a transonic business jet configuration.

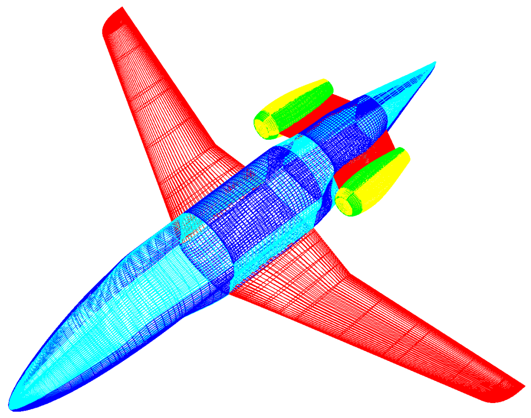

Figure 2-4: Intersected components forming the outer-mold line (OML).

displacements fields will be continuous everywhere on the surface. Although the OML is clearly defined, the interface between a typical CFD mesh and a CSM finite-element model is usually more blurry. While the CFD mesh points at the solid boundary are positioned exactly on the OML, most of the CSM finite element nodes are usually located in the interior of the volume defined by the OML. For example, wing structural models often only model the wingbox, neglecting the leading and trailing edges which do not contribute significantly to the overall stiffness of the wing.

To address these issues, we follow the work of Brown [13] in order to carry out the bidirectional transfer of loads and displacements between the CSM finite element model and the CFD mesh via the OML database.

The underlying assumption is that the mesh resolution of the *Aerosurf* database is comparable to, if not better than, that of the CFD surface mesh. This has always been the case in our design efforts. An example of an *Aerosurf* patch — with a reduced number of points, for clarity — is shown in Figure 2-6, with the surface CFD mesh points associated with the same patch. Since *Aerosurf* creates a parametric description of the surface, the patch provides a very accurate description of the real geometry. For a given geometry, and after intersecting the aircraft components, an arbitrary number of points is generated on the OML and their parametric coordinates and patch numbers are stored in the geometry database. In the next two sections we will see that the OML points play an important role in the load and displacements transfer since they have both an aerodynamic pressure and a displacement associated with them.

The CFD surface mesh points shown in Figure 2-6 lie on the OML surface and, as previously mentioned, form a grid that is usually coarser than the OML grid. The parametric coordinates of each CFD surface mesh point for all patches is also stored in the geometry database. This information is necessary for both the displacement and load transfers.

2.3.2 Displacement Transfer

The objective of the load transfer procedure is to accurately translate the nodal displacements of the CSM model to CFD mesh point displacements. The displacements calculated by the CSM solver are first transferred onto the OML grid, and then onto the CFD surface mesh.

From the CSM nodal displacements $\bar{\boldsymbol{u}}$ one can easily interpolate the displacement at any point $\boldsymbol{u}(\boldsymbol{x})$ on a given finite element using equation (2.12). For the specific case of the plate elements we use, $\boldsymbol{\eta}(\boldsymbol{x})$ is a linear interpolation function.

To determine the deflected shape of the OML from the CSM model displacement field we use a method first described by Brown [13] which rely on extrapolation functions for the displacements of the internal structure to obtain the OML displacements. These extrapolation functions are not arbitrary as they must satisfy at least two conditions.

The first condition is that, just like interpolation functions, the extrapolation functions must accurately reproduce a rigid body translation or rotation. This means that for a given set of nodal displacements corresponding to a rigid body mode, the extrapolation must yield a rigid body displacement of the OML. This is not only a reasonable requirement, but is also crucial to ensure the net force balance of the consistent load vector discussed in Section 2.3.3. The second requirement is that the resulting OML displacement field must be continuous over the whole surface.

To extrapolate the structural displacement field to the OML, each OML point, $\boldsymbol{x_o}$, is associated with a point on the structural model, $\boldsymbol{x_a}$, as shown in Figure 2-5. The associated point is such that the distance between the two points is minimized. This association is executed at the pre-processing stage and it remains the same for a given aircraft configuration, even after perturbations of the design variables.

We now assume that the vector defining the "link" between the two points, $\boldsymbol{r} = \boldsymbol{x_o} - \boldsymbol{x_a}$, maintains its position and orientation relative to the finite element that contains the associated point. The displacement of the OML point, $\boldsymbol{u_o}$, can then be written as a function of

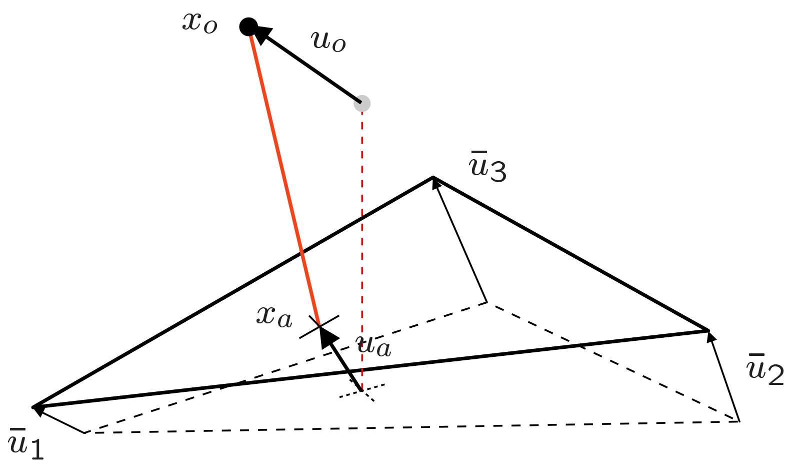

Figure 2-5: Displacement extrapolation procedure.

the associated point displacements, $\boldsymbol{u_a}$, and rotations, $\boldsymbol{\theta_a}$, i.e.,

$$\boldsymbol{u_o} = \boldsymbol{u_a} - (\boldsymbol{x_o} - \boldsymbol{x_a}) \times \boldsymbol{\theta_a} \tag{2.18}$$

$$\boldsymbol{\theta_0} = \boldsymbol{\theta_a} \tag{2.19}$$

Since the displacements and rotations at any point in a finite element can be written as a linear functions of the nodal displacements $\boldsymbol{u}$ using equation (2.12), we can rewrite the OML point displacement as,

$$\boldsymbol{u_o} = \boldsymbol{\eta}^T(\boldsymbol{x_a})\,\boldsymbol{u} - (\boldsymbol{x_o} - \boldsymbol{x_a}) \times \boldsymbol{\eta_\theta}^T(\boldsymbol{x_a})\,\boldsymbol{u}, \tag{2.20}$$

$$\boldsymbol{\theta_o} = \boldsymbol{\eta_\theta}^T(\boldsymbol{x_a})\,\boldsymbol{u}\,. \tag{2.21}$$

These equations can be re-written as,

$$\boldsymbol{u_o} = \boldsymbol{N}^T\,\boldsymbol{u}, \tag{2.22}$$

$$\boldsymbol{\theta_o} = \boldsymbol{N_\theta}^T\,\boldsymbol{u}, \tag{2.23}$$

were the extrapolation matrices are defined as,

$$\boldsymbol{N}^T = \boldsymbol{\eta}^T(\boldsymbol{x_a}) - (\boldsymbol{x_o} - \boldsymbol{x_a}) \times \boldsymbol{\eta_\theta}^T(\boldsymbol{x_a}), \tag{2.24}$$

$$\boldsymbol{N_\theta}^T = \boldsymbol{\eta_\theta}^T(\boldsymbol{x_a}). \tag{2.25}$$

Once we have the displacements for each OML point, the OML displacement field can be

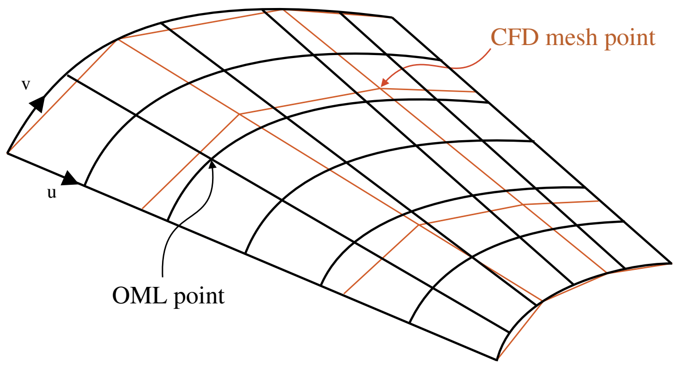

Figure 2-6: OML and CFD surface meshes on an *Aerosurf* patch.

obtained by interpolating between the points using the OML parametric description stored in the geometry database.

This means that the mapping from the OML to the finite-element model is performed on an explicit point by point basis, for a finite number of points. Displacement field continuity in the OML is then enforced directly, without requiring continuity from the underlying structural model.

Unlike the nodes of the CSM model, the CFD surface mesh points are assumed to exist on the OML. Figure 2-6 shows a representation of both the OML and CFD meshes. The parametric coordinates of the CFD surface mesh points on the corresponding OML patches are calculated in a pre-processing step via closest point projection. Therefore the patch number and the parametric coordinates of the associated point uniquely define the transfer operator. The CFD points are assumed to be "tied" to these parametric locations and any displacement of the OML, due to either design variable perturbations or structural displacements, is transferred to the CFD surface mesh points by evaluating their parametric locations on the corresponding *Aerosurf* patches.

Once a perturbation is applied to the surface of the CFD mesh, it must be propagated throughout the whole multiblock mesh. This volume mesh perturbation is achieved very efficiently by using the WARPMB algorithm. WARPMB [55] perturbs the volume mesh in four stages and is illustrated in Figure 2-7. The procedure is as follows:

1. Block faces directly affected by the surface mesh movement — the active faces — are explicitly perturbed.

2. Edges with end points in contact with active faces — in the same or in adjacent blocks — are implicitly perturbed with an arc-length attenuation method.

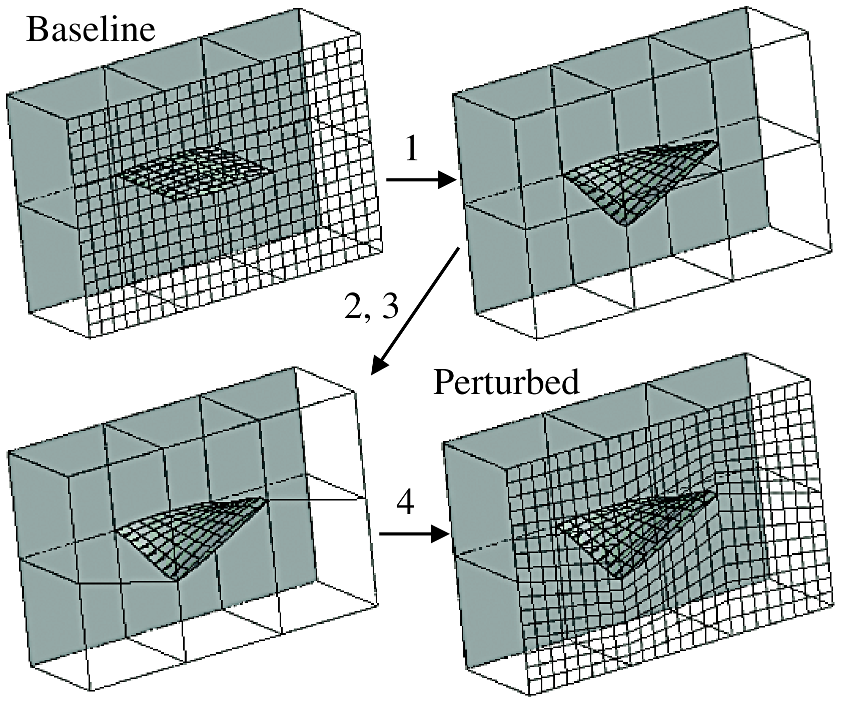

Figure 2-7: Mesh perturbation procedure used by WARPMB.

3. The interiors of any faces that are bordered by any implicitly perturbed edges or share any common edges with adjacent active faces are implicitly perturbed with WARP3QD, a quasi-three-dimensional in-plane mesh perturbation algorithm.

4. A final routine, WARP3D, is used to perturb the interiors of any blocks that have at least one active or perturbed face.

2.3.3 Load Transfer

In a similar fashion to the displacement transfer procedure, the pressures calculated by the CFD algorithm are transferred to the structural nodes through the OML points.

In order to transfer pressures from the CFD surface mesh to the OML points, we identify, in a pre-processing step, the appropriate "donor cell" and the parametric location of each OML point within this cell. The pressures at the OML points are then calculated by using bilinear interpolation on the surface of the CFD mesh. The underlying assumption that ensures the accuracy of this simple transfer is that the OML mesh is of comparable or better fidelity than that of the CFD surface mesh, and that the two surface representations are consistent and watertight.

When a distributed pressure load is applied to a structural finite-element model, it must first be transformed into an equivalent set of nodal forces. There are two requirements for this transformation, the first and more obvious of which is that the resultant nodal forces and moments be the same as those that result from the pressure field for each element. This is referred to as the *consistency* requirement and there are an infinite number of sets of nodal forces that satisfy this requirement.

However, we also require that the load transfer be *conservative*. Conservation stipulates that the virtual work performed by the load vector, $\boldsymbol{f}$, undergoing a virtual displacement of the structural model, $\delta\,\boldsymbol{u}$, must be equal to the work performed by the distributed pressure field, p, undergoing the equivalent displacement of the OML mesh, $\boldsymbol{u_o}$. The virtual work in the CSM model is given by the dot product

$$\delta W_{\text{CSM}} = \boldsymbol{f}\,\delta\,\boldsymbol{u}, \tag{2.26}$$

while the virtual work performed by the fluid acting on the surface of the OML mesh is given by the surface integral

$$\delta W_{\text{CFD}} = \int p\ \boldsymbol{n}\,\boldsymbol{u_o}\ \, \mathrm{d}S, \tag{2.27}$$

where the integral is taken over the entire OML and $\boldsymbol{n}$ represents the the unit vector normal to the OML. For a conservative scheme, $\delta W_{\text{CFD}} = \delta W_{\text{CSM}}$, and a consistent and conservative load vector then is given by

$$\boldsymbol{f} = \int p\ \boldsymbol{n}\,\boldsymbol{N}\ \mathrm{d}S, \tag{2.28}$$

where we used the linear relationship (2.23,2.23) for the virtual displacements $\boldsymbol{u_o}$. In Figure 2-8 we can see how a distributed pressure field (which has been transfered from the CFD mesh to the points on the OML) is integrated to produce a force vector that is translated into the nodal forces of a CSM element using equation (2.28).

As mentioned in Section 2.3.2, the transfer matrix $\boldsymbol{N}$ is calculated in a pre-processing step. Note that this matrix plays a dual role: it provides the appropriate weighting factors for both the transfer of OML pressures to CSM load vectors (2.28) and the transfer of the CSM displacements to OML point displacement (2.23,2.23).

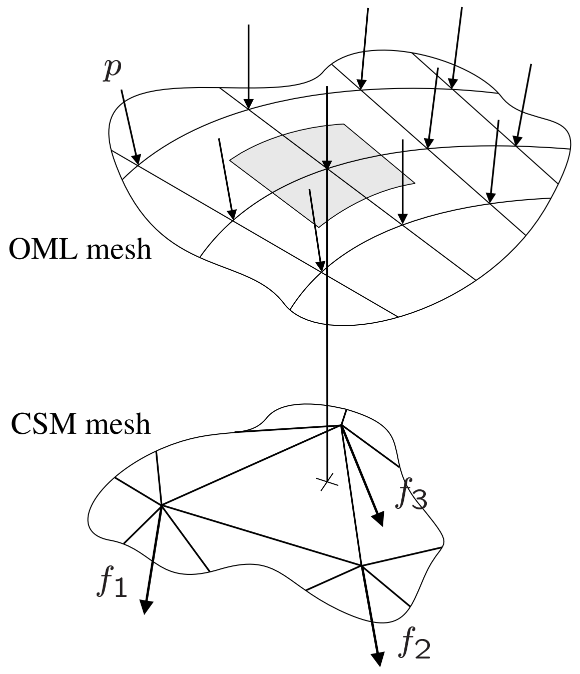

Figure 2-8: Transfer of the pressure on the OML points to the nodal forces on a given finite element

2.3.4 Aero-Structural Iteration

The aerodynamic and structural solvers are coupled by exchanging information at regular intervals during the convergence process. This coupling is greatly simplified by the fact that we only consider static aeroelastic solutions, and hence time accuracy is not an issue.

A diagram representing the aero-structural iteration is shown in Figure 2-9. The first time the flow solver is called, the displacement field of the structure is initialized to zero. After N iterations of the flow solver, the surface pressures are translated into nodal forces and the structural solver is called. The new displacement field is then translated to a movement of the CFD mesh and N more flow solver iterations are performed. The process continues until the state of the flow and the structure have converged as determined by the norm of the flow solver and structural displacement residuals. In our work, N is typically equal to 10 iterations.

For the configuration shown in Figure 4-3, running the aero-structural solver in Euler mode requires 86 multigrid cycles to reduce the average density residual by five orders of magnitude. This represents only a 15% increase when compared with the number of multigrid cycles that are required for a rigid calculation.

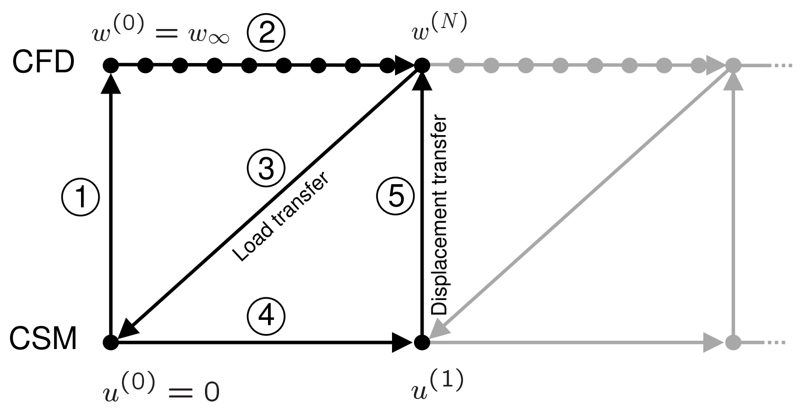

Figure 2-9: Schematic representation of the aero-structural iteration procedure.

Another factor that must be considered when comparing the cost of an aero-structural solution to an aerodynamics-only solution is the computational cost incurred by the structural solver. For the linear finite-element models we use, most of this cost is due to the factorization of the stiffness matrix. However, since for a linear system the stiffness matrix does not change unless the structure is modified, only one factorization is necessary for each aero-structural solution. During the aero-structural iteration the load vector changes periodically, and the displacement field can be quickly updated in a back-solve operation.

In cases with hundreds of thousands of degrees of freedom, for which it is impractical to factorize the stiffness matrix explicitly, the cost of the structural solution becomes significant, and we would have to resort to efficient solution methods for multiple right-hand sides.

Chapter 3

The Complex-Step Derivative Approximation

The complex-step derivative approximation is a very simple and elegant formula that can be used even in the most intricate numerical algorithm to estimate sensitivities. The computational cost of this method, like that of finite differencing, is directly proportional to the number of design variables and therefore — due to the large number of design variables encountered in the present work — the analytic method described in Chapter 4 is preferred for performing optimization. However, the complex-step approximation is still extremely useful in providing reliable benchmark results that are used for evaluating the accuracy of the analytic methods developed in the next chapter.

The objective of this chapter is to shed new light on the theory behind the complex-step derivative approximation and to show that it can be used in any algorithm that relies on real arithmetic. We also show how the complex-step method is related to algorithmic differentiation, further contributing to the understanding of this relatively new method. On the implementation side, we focus on developing *automatic* implementations, discussing the trade-offs between complex-step method and algorithmic differentiation when programming in Fortran. Finally, computational results corresponding to the application of these tools to large-scale algorithms are presented and compared with finite-difference estimates.

3.1 Background

The use of complex variables to develop estimates of derivatives originated with the work of Lyness [40] and Lyness and Moler [39]. Their papers introduced several methods that made use of complex variables, including a reliable method for calculating the n^{th} derivative of an analytic function. This theory was used by Squire and Trapp [66] to obtain a very simple expression for estimating the first derivative. This estimate is suitable for use in modern numerical computing and has been shown to be very accurate, extremely robust and surprisingly easy to implement, while retaining a reasonable computational cost. The potential of this technique is now starting to be recognized and it has been used for sensitivity analysis in CFD by Anderson et al. [5] and in an MDO environment by Newman et al. [51]. Further research on the subject has been carried out by the author [46, 47].

3.2 Theory

3.2.1 Analyticity

With real numbers alone, it is impossible to solve $x^2 = -1$. If we want to solve this equation, we must extend the set of real numbers by defining a set of new numbers

$$z = x + iy, \tag{3.1}$$

where x and y are real and $i = \sqrt{-1}$. This defines the set of complex numbers which not only enables us to solve $x^2 = -1$, but in fact any polynomial equation of degree n.

A complex-valued function $f(z)$ is said to be analytic in a given domain if it has a derivative at every point in that domain. The derivative of $f(z)$ at z is given by

$$f'(z) = \lim_{\Delta z \to 0} \frac{f(z + \Delta z) - f(z)}{\Delta z}, \tag{3.2}$$

provided this limit exists. Given this definition, we should obtain the same derivative for any small Δz. Therefore, the derivative in the direction of the real axis (i.e., for $\Delta z = \Delta x$) and the derivative in the direction of the imaginary axis (when $\Delta z = \Delta y$) must be the same, and we can write

$$\frac{\partial f}{\partial y} = i \frac{\partial f}{\partial x} \tag{3.3}$$

The real and imaginary parts of the function can be separated, i.e.,

$$f = u + iv. \tag{3.4}$$

Comparing the real and imaginary parts of equation (3.3) we obtain the familiar Cauchy–Riemann equations,

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \tag{3.5}$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}. \tag{3.6}$$

A complex function that satisfies these equations is said to be *analytic*, which is to say that it is differentiable in the complex plane.

Note that a complex number is really just *one number*. It is not unusual to forget this and think instead of a complex number as two numbers just because we do not have a way of representing it with a single axis or by a single term. This is an important concept that is useful in explaining some of the implementation issues described later in this chapter.

3.2.2 First-Derivative Approximations

Finite-differencing formulae are a common method for estimating the value of derivatives. These formulae can be derived by truncating a Taylor series which has been expanded about a given point x. A common estimate for the first derivative is the forward-difference formula, which is derived from the Taylor series expansion,

$$f(x+h) = f(x) + hf'(x) + h^2\frac{f''(x)}{2!} + h^3\frac{f'''(x)}{3!} + \ldots. \tag{3.7}$$

By solving for the first derivative, $f'(x)$ we obtain,

$$f'(x) = \frac{f(x+h) - f(x)}{h} + h\frac{f''(x)}{2!} + \ldots, \tag{3.8}$$

where h is the finite-difference interval. The truncation error is $\mathcal{O}(h)$, and therefore this represents a first-order approximation. For a second-order estimate the we can take the difference between two different Taylor series expansions of $f(x+h)$ and $f(x-h)$, to obtain the central-difference formula,

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - h^2\frac{f'''(x)}{3!} - \ldots. \tag{3.9}$$

Higher order finite-difference approximations can also be derived by using combinations of alternate Taylor series expansions.

When estimating sensitivities using finite-difference formulae we are faced with the

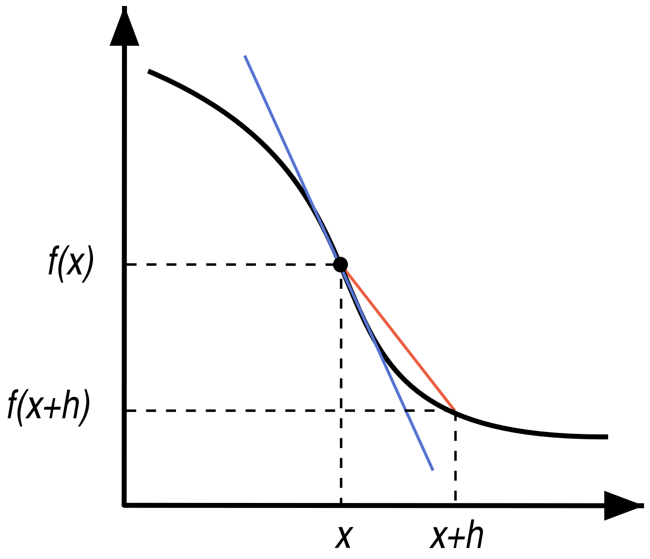

Figure 3-1: Exact and forward-finite-difference slopes of $f(x)$.

"step-size dilemma", i.e. the desire to choose a small step size to minimize truncation error while avoiding the use of a step so small that errors due to subtractive cancellation become dominant [69, 18]. The truncation error for the forward-difference approximation (3.8), for example, decreases linearly with h. Figure 3-1 shows how the estimate given by the forward difference differs from the exact slope due to the truncation error. In theory, as h tends to zero, the approximate slope tends to the exact one. However, as h tends to zero, $f(x+h)$ will have progressively more digits in common with $f(x)$, so when using finite-precision arithmetic, the accuracy of $f(x+h) - f(x)$ decreases. This phenomenon is clearly illustrated in the example of Section 3.2.3.

We now show that an equally simple first derivative estimate for real functions can be obtained using complex calculus. Consider a function, $f = u + iv$, of the complex variable, $z = x + iy$. If f is analytic the Cauchy–Riemann equations apply, i.e.,

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \tag{3.10}$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}. \tag{3.11}$$

These equations establish the exact relationship between the real and imaginary parts of the function. We can use the definition of a derivative in the right hand side of the first

Cauchy–Riemann equation (3.10) to obtain

$$\frac{\partial u}{\partial x} = \lim_{h\to 0} \frac{v(x+i(y+h)) - v(x+iy)}{h}, \tag{3.12}$$

where h is a real number. Since the functions that we are interested in are real functions of a real variable, $y = 0$, $u(x) = f(x)$ and $v(x) = 0$. The limit (3.12) can then be rewritten as

$$\frac{\partial f}{\partial x} = \lim_{h\to 0} \frac{\mathrm{Im}\left[f\left(x+ih\right)\right]}{h}. \tag{3.13}$$

For a small discrete h, this can be approximated by

$$\frac{\partial f}{\partial x} \approx \frac{\mathrm{Im}\left[f\left(x+ih\right)\right]}{h}. \tag{3.14}$$

We call this the *complex-step derivative approximation*. This estimate is not subject to subtractive cancellation error, since it does not involve a difference operation. This constitutes a tremendous advantage over the finite-difference approximation (3.8).

In order to determine the error involved in this approximation, we repeat the derivation by Squire and Trapp [66] which is based on a Taylor series expansion. Rather than using a real step h, to derive the complex-step derivative approximation we use a pure imaginary step, ih. If f is a real function in real variables and it is also analytic, we can expand it in a Taylor series about a real point x as follows,

$$f(x+ih) = f(x) + ihf'(x) - h^2\frac{f''(x)}{2!} - ih^3\frac{f'''(x)}{3!} + \ldots \tag{3.15}$$

Taking the imaginary parts of both sides of this Taylor series expansion (3.15) and dividing it by h yields

$$f'(x) = \frac{\mathrm{Im}\left[f(x+ih)\right]}{h} + h^2\frac{f'''(x)}{3!} + \ldots. \tag{3.16}$$

Hence the approximation is an $\mathcal{O}(h^2)$ estimate of the derivative of f. Notice that if we take the real part of the Taylor series expansion (3.15), we obtain the value of the function on the real axis, i.e.,

$$f(x) = \mathrm{Re}\left[f(x+ih)\right] + h^2\frac{f''(x)}{2!} - \ldots, \tag{3.17}$$

showing that $f(x)$ is also correct to $\mathcal{O}(h^2)$.

The second order errors in the function value (3.17) and the function derivative (3.15) can be eliminated when using finite-precision arithmetic by ensuring that h is sufficiently small. If ε is the relative working precision of a given algorithm, we need an h such that

$$h^2 \left| \frac{f''(x)}{2!} \right| < \varepsilon \left| f(x) \right|, \tag{3.18}$$

to eliminate the truncation error of $f(x)$ in the expansion (3.17). Similarly, for the truncation error of the derivative estimate to vanish we require that

$$h^2 \left| \frac{f'''(x)}{3!} \right| < \varepsilon \left| f'(x) \right|. \tag{3.19}$$

Although the step h can be set to extremely small values — as shown in Section 3.2.3 — it is not always possible to satisfy these conditions (3.18, 3.19), specially when $f(x), f'(x) = 0$.

3.2.3 A Simple Numerical Example

Since the complex-step approximation does not involve a difference operation, we can choose extremely small steps sizes with no loss of accuracy.

To illustrate this point, consider the following analytic function:

$$f(x) = \frac{\mathrm{e}^x}{\sqrt{sin^3 x + cos^3 x}}. \tag{3.20}$$

The exact derivative at $x = 1.5$ is computed analytically to 16 digits and then compared to the results given by the complex-step formula (3.14) and the forward and central finite-difference approximations.

Figure 3-2 shows that the forward-difference estimate initially converges to the exact result at a linear rate since its truncation error is $\mathcal{O}(h)$, while the central-difference converges quadratically, as expected. However, as the step is reduced below a value of about 10^{-8} for the forward-difference and 10^{-5} for the central difference, subtractive cancellation errors become significant and the resulting estimates are unreliable. When the interval h is so small that no difference exists in the output (for steps smaller than 10^{-16}) the finite-difference estimates eventually yields zero and then $\varepsilon = 1$.

The complex-step estimate converges quadratically with decreasing step size, as predicted by the truncation error estimate. The estimate is practically insensitive to small step sizes and for any step size below 10^{-8} it achieves the accuracy of the function evaluation. Comparing the optimum accuracy of each of these approaches, we can see that by using finite differences we only achieve a fraction of the accuracy that is obtained by using the complex-step approximation.

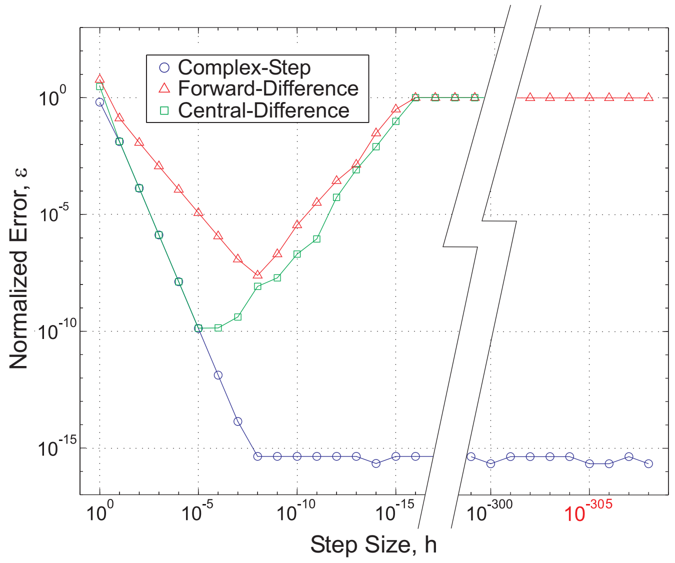

Figure 3-2: Relative error in the sensitivity estimates given by the finite-difference and the complex-step methods with the analytic result as the reference; $\varepsilon = |f' - f'_{ref}|/|f'_{ref}|$.

Although the size of the complex step can be made extremely small, there is a lower limit when using finite-precision arithmetic. The range of real numbers that can be handled in numerical computations is dependent on the particular compiler that is used. In this case, double precision arithmetic is used and the smallest non-zero number that can be represented is 10^{-308}. If a number falls below this value, underflow occurs and the representation of that number typically results in a zero value.

When comparing the relative accuracy of complex and real computations, the analysis shows that there is an increased error in basic arithmetic operations when using complex numbers, more specifically when dividing and multiplying [53].

3.2.4 Higher-Derivative Approximations

The derivative of order n of a given analytic function can be calculated by Cauchy's integral formula in its general form [61], i.e.,

$$f^{(n)}(z) = \frac{n!}{2\pi i} \int_{\Gamma} \frac{f(\xi)}{(\xi - z)^{n+1}} d\xi, \tag{3.21}$$

where Γ is a simple closed positively oriented contour that encloses z. This integral can be numerically computed using a mid-point trapezoidal rule approximation around a circle of radius r [41], yielding,

$$f^{(n)}(z) \approx \frac{n!}{mr^n} \sum_{j=0}^{m-1} \frac{f\left(z + r\,\mathrm{e}^{2\pi ij/m}\right)}{\mathrm{e}^{2\pi ijn/m}}, \tag{3.22}$$

where if m is the number of points used in the integration, we can approximate a derivative of order $n = 0, 1, \ldots, m-1$.

When comparing conventional finite-difference formulas with the complex numerical integral (3.22), we observe that both use approximations of the type $\sum a_i f(x_i)$ where the coefficients have different signs. However, there is a significant difference between the two. In conventional methods the step h has to be decreased in order to reduce the truncation error of the approximation, making it susceptible to subtractive cancellation. If we want to reduce the truncation error of the complex integration method all we need to do is to increase the number of function evaluations, i.e. m in the n^{th} derivative approximation (3.22). This keeps the subtractive cancellation error constant and it is then possible to calculate a bound on the error involved in this approximation [40].

The complex-step first derivative approximation (3.14) is a special case of the general approximation (3.22). In this special case, we are interested in the derivative of f on the real axis, i.e., when $z = x$, and we substitute r by h. By setting $m = 2$ in equation (3.22) and adding $\pi/2$ to the angle to start the integration at the top of the circle $(x + h\,\mathrm{e}^{i\pi/2})$ rather than on the right side $(x + h)$ we obtain,

$$f'(x) \approx \frac{1}{2h} \sum_{j=0}^{1} f\left(x + h\,\mathrm{e}^{i(\pi j + \pi/2)}\right) \mathrm{e}^{-i(\pi j + \pi/2)} \tag{3.23}$$

$$\approx \frac{i}{2h} \left[f(x - ih) - f(x + ih) \right]. \tag{3.24}$$

From complex variable theory, for a real function of the real variable that is analytic,

$$f(x + iy) = u + iv \Rightarrow f(x - iy) = u - iv. \tag{3.25}$$

Therefore, the approximation (3.24) can be simplified to

$$f'(x) \approx \frac{i}{2h} \left[u - iv - u - iv \right] = \frac{v}{h} \tag{3.26}$$

$$\approx \frac{\operatorname{Im}\left[f(x + ih)\right]}{h}, \tag{3.27}$$

which is identical to the second-order approximation that we previously derived (3.16). This is the only approximation that can be obtained from the n^{th} derivative approximation that does not involve subtraction and it is only valid for functions whose imaginary part is zero on the real axis.

3.2.5 Function and Operator Definitions

In the derivation of the complex-step derivative approximation for a function f (3.14) we assume that f is an analytic function, i.e. that the Cauchy–Riemann equations (3.10,3.11) apply. It is therefore important to determine to what extent this assumption holds when the value of the function is calculated by a numerical algorithm. In addition, it would be useful to explain how real functions and operators can be defined such that the complex-step derivative approximation yields the correct result when used in a computer program.

Any computer program can be broken down into a sequence of basic operations. The two main types of operations which are relevant when converting a real algorithm to a complex one are those performed by relational operators and arithmetic functions and operators.

Relational logic operators such as “greater than” and “less than” are usually not defined for complex numbers. These operators are often used in programs together with conditional statements in order to redirect the execution thread. The original algorithm and its “complexified” version must obviously follow the same execution thread. Therefore, defining these operators to compare only the real parts of the arguments is the correct approach. Functions that choose one argument such as the maximum or the minimum values are based on relational operators. Therefore, following the previous argument, we should once more choose a number based on its real part alone.

Any algorithm that uses conditional statements is likely to be a discontinuous function of its inputs. Either the function value itself is discontinuous or the discontinuity is in the first or higher derivatives. When using a finite-difference method, the derivative estimate will be incorrect if the two function evaluations are within h of the discontinuity location. However, if the complex step is used, the resulting derivative estimate will be correct right up to the discontinuity. At the discontinuity, a derivative does not exist by definition, but if the function is continuous up to that point, the approximation will still return a value corresponding to the one-sided derivative. The same is true for points where a given function has singularities.

Arithmetic functions and operators include addition, multiplication, and trigonometric functions, to name only a few. Most of these have a standard complex definition that is analytic, in which case the complex-step derivative approximation will yield the correct result.

In the case of a function that is not defined at a given point, the algorithm will not return a function value, and therefore a derivative cannot be obtained. The derivative estimate will still be correct in an arbitrary neighborhood of the discontinuity.

The only standard complex function definition that is non-analytic is the absolute value function or modulus. When the argument of this function is a complex number, the function returns the positive real number, $|z| = \sqrt{x^2 + y^2}$. The definition of this function was not derived by imposing analyticity and therefore it will not yield the correct sensitivity when using the complex-step estimate. In order to derive an analytic definition of the absolute value function we must ensure that the Cauchy–Riemann equations (3.10, 3.11) are satisfied. Since we know the exact value of the derivative, we can write

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} = \begin{cases} -1, & \text{if} \quad x < 0, \\ +1, & \text{if} \quad x > 0. \end{cases} \tag{3.28}$$

From equation (3.11), since $\partial v / \partial x = 0$ on the real axis, we get that $\partial u / \partial y = 0$ on the same axis, and therefore the real part of the result must be independent of the imaginary part of the variable. Therefore, the new sign of the imaginary part depends only on the sign of the real part of the complex number, and an analytic "absolute value" function can be defined as

$$\texttt{abs}(x + iy) = \begin{cases} -x - iy, & \text{if} \quad x < 0, \\ +x + iy, & \text{if} \quad x > 0. \end{cases} \tag{3.29}$$

Note that this is not analytic at $x = 0$ since a derivative does not exist for the real absolute value. In practice, the $x > 0$ condition is substituted by $x \geq 0$ so that we can obtain a function value for $x = 0$ and calculate the correct right-hand-side derivative at that point.

3.2.6 The Connection to Algorithmic Differentiation

When using the complex-step derivative approximation, in order to effectively eliminate truncation errors, it is typical to use a step that is many orders of magnitude smaller than the real part of the calculation. When the truncation errors are eliminated, the higher order terms of the derivative approximation (3.16) are so small that they vanish when they are added to other terms using finite-precision arithmetic.

We now observe that by linearizing the Taylor series expansion (3.15) of a complex function about x we obtain

$$f(x + ih) \equiv f(x) + ih \frac{\partial f(x)}{\partial x}, \tag{3.30}$$

where the imaginary part is exactly the derivative of f times h. The end result is a sensitivity calculation method that is equivalent to the forward mode of algorithmic differentiation, as observed by Griewank [24].

Algorithmic differentiation (AD) is a well established method for estimating derivatives [25, 9]. The method is based on the application of the chain rule of differentiation to each operation in the program flow. For each intermediate variable in the algorithm, a variation due to one input variable is carried through. As a simple example, suppose we want to differentiate the multiplication operation, $f = x_1 x_2$, with respect to x_1. Table 3.1 compares how the differentiation would be performed using either algorithmic differentiation in forward mode or the complex-step method.

Forward AD	Complex-Step Method
$\Delta x_1 = 1$	$h_1 = 10^{-20}$
$\Delta x_2 = 0$	$h_2 = 0$
$f = x_1 x_2$	$f = (x_1 + ih_1)(x_2 + ih_2)$
$\Delta f = x_1 \Delta x_2 + x_2 \Delta x_1$	$f = x_1 x_2 - h_1 h_2 + i(x_1 h_2 + x_2 h_1)$
$\partial f/\partial x_1 = \Delta f$	$\partial f/\partial x_1 = \operatorname{Im} f/h_1$

Table 3.1: The differentiation of the multiplication operation $f = x_1 x_2$ with respect to x_1 using algorithmic differentiation in forward mode and the complex-step derivative approximation.

As we can see, algorithmic differentiation stores the derivative value in a separate set of variables while the complex-step method carries the derivative information in the imaginary part of the variables. In the case of this operation, we observe that the complex-step procedure performs one additional operation — the calculation of the term $h_1 h_2$ — which, for the purposes of calculating the derivative is superfluous (and equal to zero in this case). The complex-step method will nearly always include these superfluous computations. The additional computations correspond to the higher order terms in equation (3.16).

Although this example involves only one operation, both methods work for an algorithm with an arbitrary sequence of operations by propagating the variation of one input throughout the code. This means that the cost of calculating a given set of sensitivities is proportional to the number of inputs. This particular form of algorithmic differentiation is called the *forward mode*. It calculates the derivatives of all the outputs with respect to one input. The alternative — the *reverse mode* — has no equivalent in the complex-step method.

Since the use of the complex-step method has only recently become widespread, there are some issues that seem unresolved. However, now that this connection to algorithmic differentiation is established, we can look at the extensive research on the subject of algorithmic differentiation for some answers.

Important issues include how to treat singularities [10], differentiability problems due to `if` statements [7], and the convergence of iterative solvers [6, 26], all of which have been addressed by the algorithmic differentiation research community.

The singularity issue — i.e. what to do when the derivative is infinite — is handled automatically by the complex-step method, at the expense of some accuracy. For example, the computation of $\sqrt{x+ih}$ differs substantially from $\sqrt{x}+ih/2\sqrt{x}$ as x vanishes, but this has not produced noticeable errors in the algorithms that we tested.

Regarding the issue of `if` statements, in rare circumstances, modification of the original algorithm is necessary as its differentiability may be compromised by piece-wise function definitions.

3.3 Implementation

In this section, existing algorithmic differentiation implementations are first described and then the automatic implementation of the complex-step derivative approximation is described in detail for Fortran. Collaboration with another researcher has lead to an additional implementation in C/C++[47] which will not be extensively described here. Some notes for other programming languages are also included.

3.3.1 Algorithmic Differentiation

There are two main methods for implementing algorithmic differentiation: by source code transformation or by using derived datatypes and operator overloading.

In the implementation of algorithmic differentiation by source transformation, the source code must be processed with a parser and all the derivative calculations are introduced as additional lines of code. The resulting source code is greatly enlarged and it becomes difficult to read. This constitutes an implementation disadvantage as it becomes impractical to debug this new extended code. One has to work with the original source, and every time it is changed (or if different derivatives are desired) one must rerun the parser before compiling a new version.

In order to use derived types, we need languages that support this feature, such as Fortran 90 or C++. Using this feature, algorithmic differentiation can be implemented by creating a new structure that contains both the value of the variable and its derivative. All of the existing operators are then redefined (overloaded) for the new type. The new operator exhibits the same behavior as before for the value part of the new type, but uses the definition of the derivative of the operator to calculate the derivative portion. This results in a very elegant implementation since very few changes are required in the original program.

Fortran

Many tools for automatic algorithmic differentiation of Fortran programs exist. These tools have been extensively developed and some of them provide the user with great functionality by including the option for using the reverse mode, for calculating higher-order derivatives, or for both. Tools that use the source transformation approach include: ADIFOR [9], TAMC [19], DAFOR, GRESS [29], Odyssée [14] and PADRE2. The necessary changes to the source code are made automatically. As explained before, extending the original source code is a method that usually compromises its maintainability.

The derived datatype approach is used in the following tools: AD01 [54], ADOL-F, IMAS and OPTIMA90. Although it is in theory possible to develop a script to make the necessary changes in the source code automatically, none of these tools have this ability and the changes must be done manually.

C/C++

Well established tools for automatic algorithmic differentiation also exist for C/C++. These include include ADIC [11], an implementation mirroring ADIFOR, and ADOL-C [27], a package that uses operator overloading and can operate in the forward or reverse mode and compute higher order derivatives.

3.3.2 Complex-Step Derivative Approximation

The general procedure for the implementation of the complex-step method for an arbitrary computer program can be summarized as follows:

1. Substitute all `real` type variable declarations with `complex` declarations. It is not strictly necessary to declare *all* variables complex, but it is much easier to do so.

2. Define all functions and operators that are not defined for complex arguments.

3. Add a small complex step (e.g. $h = 1 \times 10^{-20}$) to the desired x, run the algorithm that evaluates f, and then compute $\partial f / \partial x$ using equation (3.14).

The implementation of any derivative calculation method, should be as automatic as possible. Changing the source code manually is not only an extremely tedious task, but is also likely to result in the introduction of coding errors in the program.

In Fortran 90, intrinsic functions and operators (including comparison operators) can be overloaded. This means that if a particular function or operator does not accept complex arguments, one can extend it by writing another definition that does. This feature makes it much easier to implement the complex-step method since once we overload the functions and operators, there is no need to change the function calls or conditional statements. The compiler automatically determines the argument type and chooses the correct function or operation.

The complex function and operators needed for implementing the complex-step method are defined in the `complexify` Fortran 90 module. The module can be used by any subroutine in a program, including pure Fortran 77 subroutines and it redefines all intrinsic complex functions using the formula (3.30). The intrinsic functions definitions are listed in Table 3.2. Note that operators — such as addition and multiplication — which are intrinsically defined for complex arguments cannot be redefined, according to the Fortran 90 standard. However, the complex operator definitions work well for our purposes.

The way around this restriction on redefinition would be to create a new type of variable, thus allowing the use of our own function definitions. This is the approach adopted by the algorithmic differentiation methods that use derived datatypes. The drawbacks of this approach are that there would be far more changes required to the original source code and it would no longer compatible with some old Fortran constructs.

In order to automate the implementation, a script that processes Fortran source files automatically was developed. The script inserts a statement that ensures that the complex functions module is used in every subroutine, substitutes all the real type declarations by complex ones and adds `implicit complex` statements when appropriate. The script was written in Python [38] and supports a wide range of platforms and compilers. It also makes the necessary changes to MPI-based parallel implementations and takes care of file I/O statements. The latest versions of both the script and the Fortran 90 module, are available from a dedicated web page [42].

This tool for implementing the complex-step method represents, in my opinion, a good compromise between ease of implementation and algorithmic efficiency. While pure algorithmic differentiation is numerically more efficient, the complex-step method requires far fewer changes to the original source code, due to the fact that complex variables are a Fortran intrinsic type. The end result is improved maintainability. Furthermore, practically

Fortran Function	Definition
`abs`	$abs(z) = \begin{cases} -z & \Leftarrow x < 0 \\ +z & \Leftarrow x \geq 0 \end{cases}$
`exp`	$\mathrm{e}^z = \mathrm{e}^x(\cos(y) + i\sin(y))$
`sqrt`	$\sqrt{z} = \sqrt{\lvert z\rvert}\Big(\cos(\arg(z)/2) + i\sin(\arg(z)/2)\Big)$
`sin`	$\sin(z) = \left(\mathrm{e}^{iz} - \mathrm{e}^{-iz}\right)/2i$
`cos`	$\cos(z) = \left(\mathrm{e}^{iz} + \mathrm{e}^{-iz}\right)/2$
`tan`	$\tan(z) = \tan(x) + iy/\cos^2(x)$
`log`	$\log(z) = \log\lvert z\rvert + i\arg(z)$
`log10`	$\log_{10}(z) = \log(z)/\log(10)$
`asin`	$\arcsin(z) = \arcsin(x) + iy/\left(\sqrt{1-x^2}\right)$
`acos`	$\arccos(z) = \arccos(x) - iy/\left(\sqrt{1-x^2}\right)$
`atan`	$\arctan(z) = \arctan(x) + iy/\left(\sqrt{1+x^2}\right)$
`atan2`	$\arctan2(z_1, z_2) = \arctan(x_2/x_1) + (x_2y_1 - x_1y_2)/\left(x_1^2 + x_2^2\right)$
`sinh`	$\sinh(z) = \sinh(x) + iy\cosh(x)$
`cosh`	$\cosh(z) = \cosh(x) + iy\sinh(x)$
`tanh`	$\tanh(z) = \tanh(x) + iy/\cosh^2(x)$
`dim`	$\dim(z_1, z_2) = \begin{cases} z_1 - z_2 & \Leftarrow x_1 > x_2 \\ 0 & \Leftarrow x_1 \leq x_2 \end{cases}$
`sign`	$\mathrm{sign}(z_1, z_2) = \begin{cases} +\lvert x_1\rvert & \Leftarrow x_2 \geq 0 \\ -\lvert x_1\rvert & \Leftarrow x_2 < 0 \end{cases}$
`max`	$\max(z_1, z_2) = \begin{cases} z_1 & \Leftarrow x_1 \geq x_2 \\ z_2 & \Leftarrow x_1 < x_2 \end{cases}$
`min`	$\min(z_1, z_2) = \begin{cases} z_1 & \Leftarrow x_1 \leq x_2 \\ z_2 & \Leftarrow x_1 > x_2 \end{cases}$

Table 3.2: Fortran intrinsic function definitions used in `complexify.f90`, $z = x + iy$.

all the changes are performed automatically by the use of the script.

Other Programming Languages

In addition to the Fortran and C/C++ implementations described above, some experimentation was done with other programming languages.

C/C++: Neither C nor C++ perform complex arithmetic by default, although there are complex arithmetic libraries that one can use. In C++, all operators and functions can be overloaded just as in Fortran 90. The implementation of the complex-step procedure in C/C++ is discussed in previous work [47].

Matlab: As in the case of Fortran, one must redefine functions such as `abs`, `max` and `min`. All differentiable functions are defined for complex variables. The results shown in Figure 3-2 are actually computed using Matlab. The standard transpose operation represented by an apostrophe (`'`) poses a problem as it takes the complex conjugate of the elements of the matrix, so one should use the non-conjugate transpose represented by "dot apostrophe" (`.'`) instead.

Java: Complex arithmetic is not standardized at the moment but there are plans for its implementation. Although function overloading is possible, operator overloading is currently not supported.

Python: A simple implementation of the complex-step method for Python was also developed in this work. The `cmath` module must be imported to gain access to complex arithmetic. Since Python supports operator overloading, it is possible to define complex functions and operators as described earlier.

Algorithmic differentiation can be implemented in any programming language that supports derived datatypes and operator overloading. For languages that do not have these features, the complex-step method can be used wherever complex arithmetic is supported.

3.4 Results

The tool developed in this dissertation to implement the complex-step method automatically in Fortran has been tested on a variety of programs. One of the most complicated examples is the high-fidelity aero-structural solver described in Chapter 2.

Before discussing the results from the aero-structural solver, we present a study of the accuracy and efficiency of different sensitivity analysis methods for a structural finite-element solver.

3.4.1 Structural Sensitivities

The structural solver and wing model used in this study are the ones described in Section 2.2. In this example, the structural box of the wing of a transonic transport is modeled.

In a first study, the sensitivity estimates given by the complex-step derivative approximation (3.14) and forward finite-difference (3.8) methods are compared for various step sizes. The sample sensitivity chosen for this study is that of the stress in a spar cap with respect to its own cross-sectional area. This kind of sensitivity is very important in gradient-based structural optimization, where the derivatives of the stress constraints are usually required.

Note that in this case, stress is a nonlinear function of the cross-sectional area and, therefore, we should be able to observe the rate convergence of the estimates for decreasing step sizes.

Figure 3-3 shows a comparison of results — analogous to that of Figure 3-2 — with a reference derivative value which is obtained by an exact analytic method. The analytic method used here is the direct method applied to a discrete linear set of equations as described by Adelman [1]. This method is included here only to provide a benchmark, since it has an implementation that is far more involved and code-specific than the other ones.

As expected, the error of the finite-difference estimate initially decreases at a linear rate. As the step is reduced to a value of about 10^{-6}, subtractive cancellation errors become increasingly significant and the estimate of the error increases. For even smaller perturbations — of the order of 10^{-17} — no difference exists in the output and the finite-difference estimate eventually goes to zero ($\varepsilon = 1$).

The complex-step estimate converges quadratically with decreasing step size, converging to the precision of the structural solver when h is of the order of 10^{-7}. Note that the estimate is still accurate down to a step of order 10^{-306}. Below this, underflow starts to occur, the estimate is corrupted, and eventually the result becomes meaningless.

A second study compares the accuracy and computational cost between the three methods mentioned above to an implementation of algorithmic differentiation for Fortran known as ADIFOR. This is a package that automatically processes a given Fortran program, producing a new program that in addition to the original computations also calculates the desired sensitivities. A sample of the sensitivity results is shown in Table 3.3, and a computational cost comparison is made in Table 3.4. The computational cost values are normalized

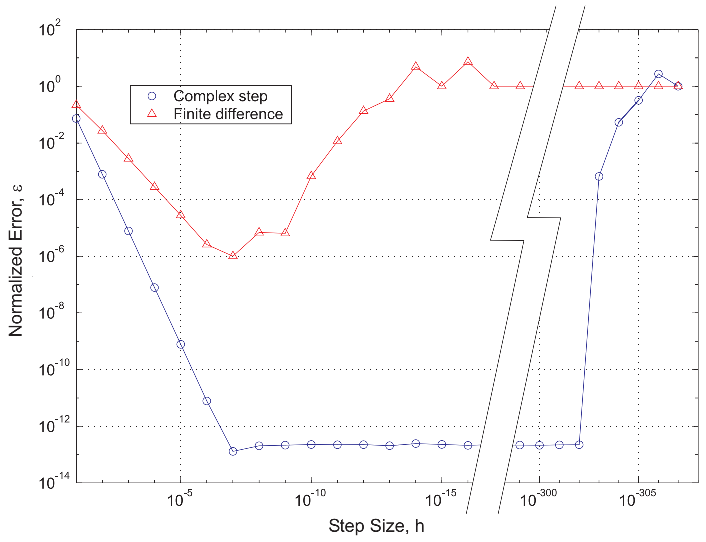

Figure 3-3: Error of sensitivity estimates given by finite difference and complex step with the analytic method result as the reference; $\varepsilon = |f' - f'_{ref}|/|f'_{ref}|$.

with respect to the computation time and memory usage of the complex-step method. The computations are performed on a SGI Octane with a 195MHz R10000 processor and correspond to the calculation of the sensitivities of the stress in all of the 60 trusses in the wing structural model with respect to their cross-sectional areas, resulting in a total of 3,600 sensitivities.

The sample sensitivity shown in Table 3.3 is the same one that was used to produce the results shown previously in Figure 3-3. When compiled using double precision, the finite-element solver has an accuracy of about 13 digits, so the last 4 digits should be ignored.

The finite-difference sensitivity estimate — shown at the bottom of Tables 3.3 and 3.4 —

Method	Sample Sensitivity
Complex	-39.04976004580*4646*
ADIFOR	-39.04976004580*9059*
Analytic	-39.04976004580*5281*
FD	-39.0497*24352820375*

Table 3.3: Sensitivity estimate accuracy comparison.

Method	Time	Memory
Complex	1.00	1.00
ADIFOR	2.33	8.09
Analytic	0.58	2.42
FD	0.88	0.72

Table 3.4: Relative computational cost comparison for the calculation of the complete Jacobian.

is obtained using an optimal step ($h = 10^{-7}$). Even then, the estimate is shown to be only half as accurate as the other ones. Although this method is extremely easy to implement, finding a step that gives reasonably accurate estimates is usually a problem and, therefore, the total computation time is in practice much higher than the one reflected by these results.

As expected, the analytic method was the most accurate, and by far the fastest. The computation using this method required considerably more memory since a number of new variables were introduced in the algorithm. As mentioned before, the implementation of the analytic method is much more involved than in the other cases and this places this method in a class of its own.

ADIFOR produced very accurate estimates but it was the most costly, with respect to both computation time and memory usage. This has to do with the fact that ADIFOR produces a code which is much larger than the original one and which contains many more statements and variables. This fact constitutes an implementation disadvantage as it becomes impractical to debug this new code.

The complex-step method was also accurate to the precision of the solver, was reasonably fast, and used less memory than any other method with the exception of the finite-difference method. The results were obtained using $h = 10^{-100}$. In general, the memory requirement will always be greater than in the case of finite differencing, but never more than twice as much. As opposed to ADIFOR, the new "complexified" code is practically identical to the original one and can therefore be worked on directly. With the help of a few compiler flags one can even produce a single code that can be chosen to be real or complex at compilation time.

Finally, note that the relative costs given in Table 3.4 may vary for different problems,

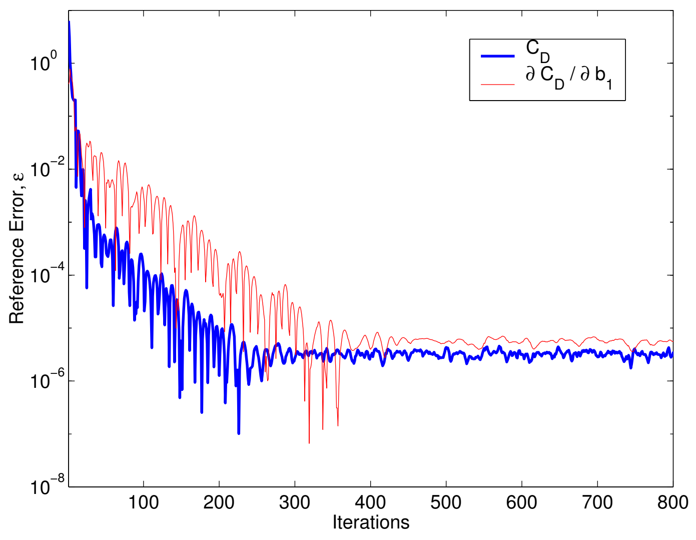

Figure 3-4: Convergence of C_D and $\partial C_D/\partial x_1$ for the aero-structural solver; $\varepsilon = |f - f_{ref}|/|f_{ref}|$.

since these costs depend heavily on the ratio of the number of outputs we want to differentiate to the number of design variables with respect to which we want to differentiate. However, the costs associated with the complex-step method will always be proportional to those of the finite-difference method.

3.4.2 Aero-Structural Sensitivities

The aero-structural solver and model used here are the same as described in Chapter 2. The subsequent results have been obtained for the isolated wing of a small transonic business jet flying at a free-stream Mach number of 0.82 and lift coefficient of 0.352.

To validate the complex-step results for the aero-structural solver, we chose the sensitivity of the drag coefficient with respect to a set of 18 wing shape perturbations.

Since the aero-structural solver is an iterative algorithm, it is useful to compare the convergence of a given function with that of its derivative, which is contained in its complex part. This comparison is shown in Figure 3-4 for the drag coefficient and its derivative with the respect to the first shape design variable, x_1. The drag coefficient converges to the precision of the algorithm in about 300 iterations. The drag sensitivity converges at the same rate as the coefficient and it lags slightly, taking about 100 additional iterations to achieve the maximum precision. This is expected, since the calculation of the sensitivity of a given quantity is dependent on the value of that quantity. The minimum error in the

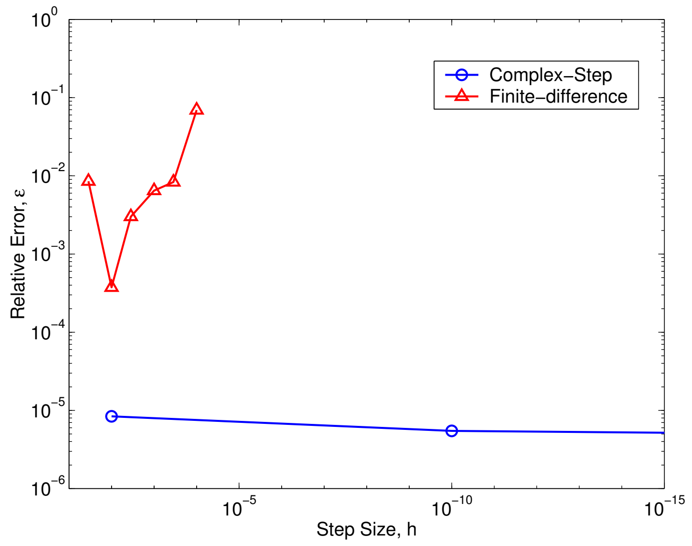

Figure 3-5: Sensitivity estimate errors for $\partial C_D/\partial x_1$ given by finite difference and the complex step for different step sizes; $\varepsilon = |f - f_{ref}|/|f_{ref}|$; reference is complex-step estimate at $h = 10^{-20}$.

derivative is observed to be slightly lower than the precision of the coefficient. When looking at the number of digits that are converged, the drag coefficient consistently converges to six digits, while the derivative converges to five or six digits. This can be explained by the increased round-off errors of complex arithmetic [53], which do not affect the real part when such small step sizes are used.

The plot shown in Figure 3-5 is analogous to that of Figure 3-2, where the sensitivity estimates given by the complex-step and forward finite-difference methods are compared for a varying step sizes. In this case the finite-difference result has an acceptable precision only for one step size ($h = 10^{-2}$). Again, the complex-step method yields accurate results for a wide range of step sizes, from $h = 10^{-2}$ to $h = 10^{-200}$ in this case.

The results corresponding to the complete shape sensitivity vector are shown in Figure 3-6. Although many different sets of finite-difference results were obtained, only the set corresponding to the optimum step is shown. The plot shows no discernible difference between the two sets of results.

A comparison of the relative computational cost of the two methods was also performed for the aerodynamic sensitivities, namely for the calculation of the complete shape sensitivity vector. Table 3.5 lists these costs, normalized with respect to the solution time of the aero-structural solver.

The cost of a finite-difference gradient evaluation for the 18 design variables is about 14

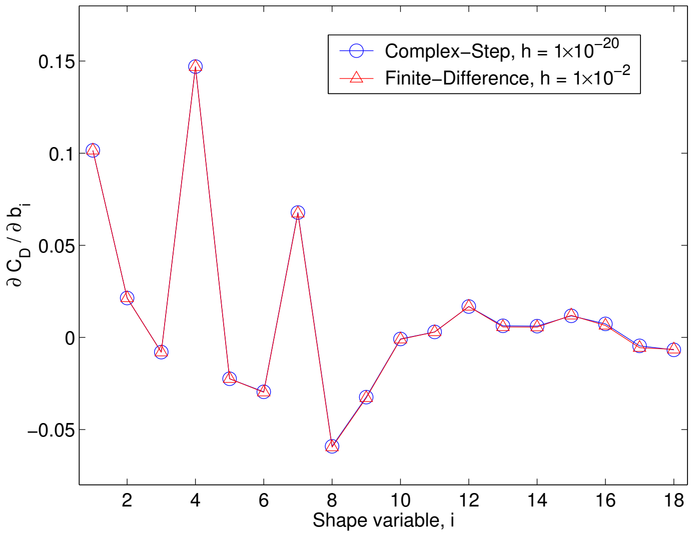

Figure 3-6: Comparison of the estimates for the shape sensitivities of the drag coefficient, $\partial C_D/\partial x_i$.

Computation Type	Normalized Cost
Aero-structural Solution	1.0
Finite difference	14.2
Complex step	34.4

Table 3.5: Normalized computational cost comparison for the calculation of the complete shape sensitivity vector.

times the cost of a single aero-structural solution for computations that have converged to six orders of magnitude in the average density residual. Notice that one would expect this method to incur a computational cost equivalent to 19 aero-structural solutions (the solution of the baseline configuration plus one flow solution for each design variable perturbation.) The cost is lower than this value because the additional calculations start from the previously converged solution.

The cost of the complex-step procedure is more than twice of that of the finite-difference procedure since the function evaluations require complex arithmetic. However, the complex-step calculations are worth this cost penalty since there is no need to find an acceptable step size *a priori*, as in the case of the finite-difference approximations: while there was considerable effort involved in obtaining reasonable finite-difference results by trying different step sizes, no such studies were necessary with the complex-step method.

Chapter 4

Analytic Sensitivity Analysis of Coupled Systems

In this chapter we present the analytic sensitivity method that is used to solve the final optimization problem of this dissertation. The following section begins with an introduction to analytic sensitivity analysis where both the direct and adjoint methods are derived. We then generalize this theory for coupled systems and derive the sensitivity equations specific to the aero-structural solver used in the present work. These equations are then solved to obtain the vector of sensitivities of the wing drag coefficient with respect to wing-shape variables and we show that, when using the coupled-adjoint method, these sensitivities can be obtained accurately and efficiently. Finally, we present results of the application of this sensitivity computation method to the aero-structural optimization of a transonic wing.

4.1 General Formulation

The main objective is to calculate the sensitivity of a multidisciplinary function of interest with respect to a number of design variables. The function of interest can be either the objective function or any of the constraints specified in the optimization problem. In general, such functions depend not only on the design variables, but also on the physical state of the multidisciplinary system. Thus we can write the function as

$$I = I(x_n, y_i), \tag{4.1}$$

where x_n represents the vector of design variables and y_i is the state variable vector.

For a given vector x_n, the solution of the governing equations of the multidisciplinary system yields a vector y_i, thus establishing the dependence of the state of the system on the design variables. We denote these governing equations by

$$\mathcal{R}_k\left(x_n, y_i\left(x_n\right)\right) = 0. \tag{4.2}$$

The first instance of x_n in the above equation indicates the fact that the residual of the governing equations may depend *explicitly* on x_n. In the case of a structural solver, for example, changing the size of an element has a direct effect on the stiffness matrix. By solving the governing equations we determine the state, y_i, which depends *implicitly* on the design variables through the solution of the system. These equations may be non-linear, in which case the usual procedure is to drive residuals, $\mathcal{R}_k$, to zero using an iterative method.

Since the number of equations must equal the number of state variables, the ranges of the indices i and k are the same, i.e., $i, k = 1, \ldots, N_{\mathcal{R}}$. In the case of a structural solver, for example, $N_{\mathcal{R}}$ is the number of degrees of freedom, while for a CFD solver, $N_{\mathcal{R}}$ is the number of mesh points multiplied by the number of state variables at each point. In the more general case of a multidisciplinary system, $\mathcal{R}_k$ represents *all* the governing equations of the different disciplines, including their coupling.

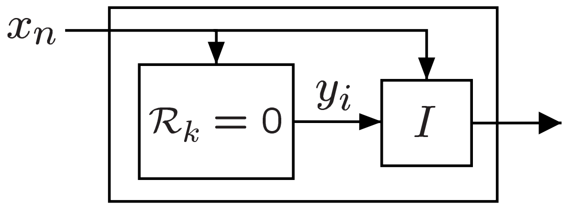

Figure 4-1: Schematic representation of the governing equations ($\mathcal{R}_k = 0$), design variables (x_n), state variables (y_i), and objective function (I), for an arbitrary system.

A graphical representation of the system of governing equations is shown in Figure 4-1, with the design variables x_n as the inputs and I as the output. The two arrows leading to I illustrate the fact that the objective function typically depends on the state variables and may also be an explicit function of the design variables.

As a first step toward obtaining the derivatives that we ultimately want to compute, we use the chain rule to write the total sensitivity of I as

$$\frac{\mathrm{d}I}{\mathrm{d}x_n} = \frac{\partial I}{\partial x_n} + \frac{\partial I}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}x_n}, \tag{4.3}$$

for $i = 1, \ldots, N_{\mathcal{R}}$, $n = 1, \ldots, N_x$. Index notation is used to denote the vector dot products. It is important to distinguish the total and partial derivatives in this equation. The partial derivatives can be directly evaluated by varying the denominator and re-evaluating the function in the numerator. The total derivatives, however, require the solution of the multidisciplinary problem. Thus, all the terms in the total sensitivity equation (4.3) are easily computed except for $\mathrm{d}y_i/\,\mathrm{d}x_n$.

Since the governing equations must always be satisfied, the total derivative of the residuals (4.2) with respect to any design variable must also be zero. Expanding the total derivative of the governing equations with respect to the design variables we can write,

$$\frac{\mathrm{d}\mathcal{R}_k}{\mathrm{d}x_n} = \frac{\partial \mathcal{R}_k}{\partial x_n} + \frac{\partial \mathcal{R}_k}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}x_n} = 0, \tag{4.4}$$

for all $i, k = 1, \ldots, N_{\mathcal{R}}$ and $n = 1, \ldots, N_x$. This expression provides the means for computing the total sensitivity of the state variables with respect to the design variables. By rewriting equation (4.4) as

$$\frac{\partial \mathcal{R}_k}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}x_n} = -\frac{\partial \mathcal{R}_k}{\partial x_n}, \tag{4.5}$$

we can solve for $\mathrm{d}y_i/\,\mathrm{d}x_n$ and substitute this result into the total derivative equation (4.3), to obtain

$$\frac{\mathrm{d}I}{\mathrm{d}x_n} = \frac{\partial I}{\partial x_n} - \underbrace{\frac{\partial I}{\partial y_i}\overbrace{\left[\frac{\partial \mathcal{R}_k}{\partial y_i}\right]^{-1}}_{-\Psi_k}\frac{\partial \mathcal{R}_k}{\partial x_n}}^{-\,\mathrm{d}y_i/\,\mathrm{d}x_n}. \tag{4.6}$$

The inversion of the Jacobian $\partial \mathcal{R}_k/\partial y_i$ is not necessarily explicitly calculated. In the case of large iterative problems neither this matrix nor its factorization are usually stored due to their prohibitive size.

The approach where we first calculate $\mathrm{d}y_i/\,\mathrm{d}x_n$ using equation (4.5) and then use the result in the expression for the total sensitivity (4.6) is called the *direct* method. Note that solving for $\mathrm{d}y_i/\,\mathrm{d}x_n$ requires the solution of the matrix equation (4.5) *for each design variable* x_n. A change in the design variable affects only the right-hand side of the equation, so for problems where the matrix $\partial \mathcal{R}_k/\partial y_i$ can be explicitly factorized and stored, solving for multiple right-hand-side vectors by back substitution would be relatively inexpensive. However, for large iterative problems — such as the ones encountered in CFD — the matrix $\partial \mathcal{R}_k/\partial y_i$ is never factorized explicitly and the system of equations requires an iterative solution which is usually as costly as solving the governing equations. When we multiply this cost by the number of design variables, the total cost for calculating the sensitivity vector may become unacceptable.

Returning to the total sensitivity equation (4.6), we observe that there is an alternative option when computing the total sensitivity $\mathrm{d}I/\mathrm{d}x_n$. The auxiliary vector Ψ_k can be obtained by solving the *adjoint equations*

$$\frac{\partial \mathcal{R}_k}{\partial y_i}\Psi_k = -\frac{\partial I}{\partial y_i}. \tag{4.7}$$

The vector Ψ_k is usually called the *adjoint vector* and is substituted into equation (4.6) to find the total sensitivity. In contrast with the direct method, the adjoint vector does not depend on the design variables, x_n, but instead depends on the function of interest, I.

We can now see that the choice of the solution procedure (direct vs. adjoint) to obtain the total sensitivity (4.6) has a substantial impact on the cost of sensitivity analysis. Although all the partial derivative terms are the same for both the direct and adjoint methods, the order of the operations is not. Notice that for any number of functions, I, we can compute $\mathrm{d}y_i/\mathrm{d}x_n$ once for each design variable (direct method). Alternatively, for an arbitrary number of design variables, we can compute Ψ_k once for each function (adjoint method).

The cost involved in calculating sensitivities using the adjoint method is therefore practically independent of the number of design variables. After having solved the governing equations, the adjoint equations are solved only once for each I. Moreover, the cost of solution of the adjoint equations is similar to that of the solution of the governing equations since they are of similar complexity and the partial derivative terms are easily computed.

Therefore, if the number of design variables is greater than the number of functions for which we seek sensitivity information, the adjoint method is computationally more efficient. Otherwise, if the number of functions to be differentiated is greater than the number of design variables, the direct method would be a better choice.

The adjoint method has been widely used for single discipline sensitivity analysis and examples of its application include structural sensitivity analysis [1] and aerodynamic shape optimization [30, 33].

4.2 Aero-Structural Sensitivity Equations

Although the theory we have just presented is applicable to multidisciplinary systems, provided that the governing equations for all disciplines are included in $\mathcal{R}_k$, we now explicitly discuss the sensitivity analysis of multidisciplinary systems, using aero-structural optimization as an example. This example illustrates the fundamental computational cost issues that motivate our choice of strategy for sensitivity analysis. The following equations and discussion can easily be generalized for cases with additional disciplines.

In the aero-structural case we have coupled aerodynamic ($\mathcal{A}_k$) and structural ($\mathcal{S}_l$) governing equations, and two sets of state variables: the flow state vector, w_i, and the vector of structural displacements, u_j. In the following expressions, we split the vectors of residuals, states and adjoints into two smaller vectors corresponding to the aerodynamic and structural systems

$$\mathcal{R}_{k'} = \begin{bmatrix} \mathcal{A}_k \\ \\ \mathcal{S}_l \end{bmatrix}, \quad y_{i'} = \begin{bmatrix} w_i \\ \\ u_j \end{bmatrix}, \quad \Psi_{k'} = \begin{bmatrix} \psi_k \\ \\ \phi_l \end{bmatrix}. \tag{4.8}$$

Figure 4-2 shows a diagram representing the coupling in this system.

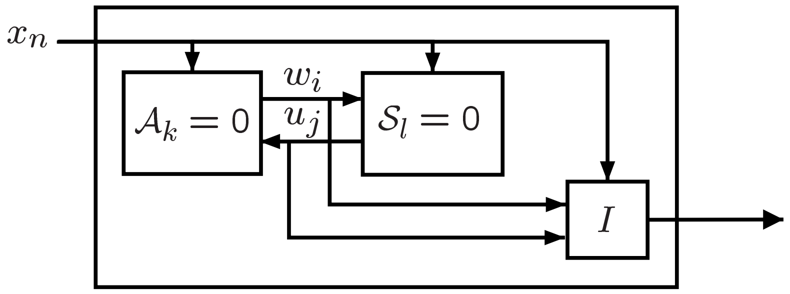

Figure 4-2: Schematic representation of the aero-structural system.

4.2.1 Coupled-Direct Methods

Using this new notation, the direct sensitivity equation (4.5) for an aero-structural system can be written as

$$\begin{bmatrix} \frac{\partial \mathcal{A}_k}{\partial w_i} & \frac{\partial \mathcal{A}_k}{\partial u_j} \\ \frac{\partial \mathcal{S}_l}{\partial w_i} & \frac{\partial \mathcal{S}_l}{\partial u_j} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d} w_i}{\mathrm{d} x_n} \\ \frac{\mathrm{d} u_j}{\mathrm{d} x_n} \end{bmatrix} = - \begin{bmatrix} \frac{\partial \mathcal{A}_k}{\partial x_n} \\ \frac{\partial \mathcal{S}_l}{\partial x_n} \end{bmatrix}. \tag{4.9}$$

This equation was first written for a multidisciplinary system by Sobieski [63].

In his paper, Sobieski also presents an alternative approach to the problem, which he shows is equivalent to (4.9), i.e.,

$$\begin{bmatrix} \mathcal{I} & -\frac{\partial w_i}{\partial u_j} \\ -\frac{\partial u_j}{\partial w_i} & \mathcal{I} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{d}w_i}{\mathrm{d}x_n} \\ \frac{\mathrm{d}u_j}{\mathrm{d}x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial w_i}{\partial x_n} \\ \frac{\partial u_j}{\partial x_n} \end{bmatrix}, \tag{4.10}$$

where $\mathcal{I}$ denotes the identity matrix. Solving either of these equations (4.9, 4.10) yields the total sensitivity of the state variables with respect to the design variables. This result can then be substituted into the aero-structural equivalent of the total sensitivity equation (4.3),

$$\frac{\mathrm{d}I}{\mathrm{d}x_n} = \frac{\partial I}{\partial x_n} + \frac{\partial I}{\partial u_j}\frac{\mathrm{d}u_j}{\mathrm{d}x_n} + \frac{\partial I}{\partial w_i}\frac{\mathrm{d}w_i}{\mathrm{d}x_n}. \tag{4.11}$$

In the alternate direct approach (4.10), the partial derivatives of the state variables of a given system with respect to the variables of the other system ($\partial w_i/\partial u_j$, $\partial u_j/\partial w_i$) and the partial derivatives of the state variables with respect to the design variables ($\partial w_i/\partial x_n$, $\partial u_j/\partial x_n$) have a different meaning from the partial derivatives we have seen so far. In this formulation, the partial derivatives of the state variables of a given system take into account the solution of that system. Although the solution of the coupled system is not required, this is in contrast with the partial derivatives of the residuals in the formulation (4.9), which do not require the solution of even the single discipline.

The greatest disadvantage of both of these direct approaches, as we discussed earlier, is that the sensitivity equation must be solved for each design variable x_n. For large iterative coupled systems, the cost of computing the total sensitivities with respect to many design variables becomes prohibitive, and this approach is impractical.

4.2.2 Coupled-Adjoint Methods

The adjoint approach to sensitivity analysis is also applicable to multidisciplinary systems. In the case of the aero-structural system, the adjoint equation (4.7) can be written as

$$\begin{bmatrix} \frac{\partial \mathcal{A}_k}{\partial w_i} & \frac{\partial \mathcal{A}_k}{\partial u_j} \\ \frac{\partial \mathcal{S}_l}{\partial w_i} & \frac{\partial \mathcal{S}_l}{\partial u_j} \end{bmatrix}^T \begin{bmatrix} \psi_k \\ \phi_l \end{bmatrix} = - \begin{bmatrix} \frac{\partial I}{\partial w_i} \\ \frac{\partial I}{\partial u_j} \end{bmatrix}. \tag{4.12}$$

Note that the matrix in the coupled adjoint equation is the same as in the coupled direct method (4.9). This matrix, in addition to containing the diagonal terms that appear when we solve the single discipline adjoint equations, also has off-diagonal terms expressing the sensitivity of one discipline to the state variables of the other. The details of the partial derivative terms of this matrix are described in Section 4.3.

Finally, for completeness, we note that there is an alternative formulation for the coupled-adjoint method which is parallel to the alternate direct equations (4.10),

$$\begin{bmatrix} \mathcal{I} & -\frac{\partial w_i}{\partial u_j} \\ -\frac{\partial u_j}{\partial w_i} & \mathcal{I} \end{bmatrix}^T \begin{bmatrix} \bar{\psi}_i \\ \bar{\phi}_j \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial w_i} \\ \frac{\partial I}{\partial u_j} \end{bmatrix}. \tag{4.13}$$

The alternate adjoint vector, $\bar{\psi}_k$ has a different significance here and the total sensitivity equation for this case is given by

$$\frac{\mathrm{d}I}{\mathrm{d}x_n} = \frac{\partial I}{\partial x_n} + \bar{\psi}_i \frac{\partial w_i}{\partial x_n} + \bar{\phi}_j \frac{\partial u_j}{\partial x_n}, \tag{4.14}$$

where the partial derivatives have the same meaning as in the alternate direct sensitivity equations (4.10) and are thus rather costly to compute.

4.2.3 The Lagged Coupled-Adjoint Equations

Since the factorization of the full matrix in the coupled-adjoint equations (4.12) would be extremely costly, our approach uses an iterative solver, much like the one used for the aero-structural solution, where the adjoint vectors are *lagged* and the two different sets of equations are solved separately. For the calculation of the adjoint vector of one discipline, we use the adjoint vector of the other discipline from the previous iteration, i.e., we solve

$$\underbrace{\frac{\partial \mathcal{A}_k}{\partial w_i}\psi_k = -\frac{\partial I}{\partial w_i}}_{\text{Aerodynamic adjoint}} - \frac{\partial \mathcal{S}_l}{\partial w_i}\tilde{\phi}_l, \tag{4.15}$$

$$\underbrace{\frac{\partial \mathcal{S}_l}{\partial u_j}\phi_l = -\frac{\partial I}{\partial u_j}}_{\text{Structural adjoint}} - \frac{\partial \mathcal{A}_k}{\partial u_j}\tilde{\psi}_k, \tag{4.16}$$

where $\tilde{\psi}_k$ and $\tilde{\phi}_l$ are the lagged aerodynamic and structural adjoint vectors respectively. Upon convergence, the final result given by this system, is the same as that given by the original coupled-adjoint equations (4.12). We call this the *lagged-coupled adjoint* (LCA) method for computing sensitivities of coupled systems. Note that these equations look like the single discipline adjoint equations for the aerodynamic and structural solvers, with the addition of forcing terms in the right-hand side that contain the off-diagonal terms of the residual sensitivity matrix. This allows us to use existing single-discipline adjoint sensitivity analysis methods. Note also that, even for more than two disciplines, this iterative solution procedure is nothing more than the well-known block-Jacobi method.

Once both adjoint vectors have converged, we can compute the final sensitivities of the objective function by using the following expression

$$\frac{\mathrm{d}I}{\mathrm{d}x_n} = \frac{\partial I}{\partial x_n} + \psi_k \frac{\partial \mathcal{A}_k}{\partial x_n} + \phi_l \frac{\partial \mathcal{S}_l}{\partial x_n}, \tag{4.17}$$

which is the coupled version of the total sensitivity equation (4.6).

4.2.4 Discussion

The approach for solving the coupled system of sensitivity equations by lagging can also be used to solve the direct equations (4.9, 4.10) but the disadvantages of these methods for problems that require iterative methods and are parameterized with a large number of design variables remain the same.

For the aero-structural optimization problem at hand the aerodynamic portion is usually characterized by a single objective function and at most a few aerodynamic constraints, but a large number of design variables. On the other hand, the structural portion of the optimization problem is characterized by a large number of constraints: the stress in each element of the finite-element model cannot exceed the material yield stress for a set of load conditions. Constrained gradient optimization methods generally require that the user provide the gradient of both the cost function and each non-linear constraint with respect to all of the design variables in the problem. Using the adjoint approach, the evaluation of the gradient of each constraint would require an independent coupled solution of a large adjoint system. Since the number of structural constraints is similar to the number of design variables in the problem ($\mathcal{O}(10^3)$ or larger), the usefulness of the adjoint approach is questionable.

Both of the remaining alternatives, the direct and finite-difference methods, are not advantageous either since they both require a number of solutions that is comparable to the number of design variables. In the absence of other choices that can efficiently evaluate the gradient of a large number of constraints with respect to a large number of design variables, it is necessary to reduce the size of the problem either through a reduction in the number of design variables or through a reduction in the number of non-linear constraints.

The reason for the choice of the KS functions to lump the structural constraints now becomes clear. By employing KS functions, the number of structural constraints for the problem can be reduced from $\mathcal{O}(10^3)$ to just a few. In some problems, a single KS function may suffice. If this constraint lumping methodology is effective, an adjoint method would be very efficient for MDO sensitivities.

4.3 Partial Derivative Term Details

In this section, the details of the calculation of the partial derivative terms in the aero-structural adjoint equations (4.15, 4.16) and the total sensitivity equations (4.17) are described. This description is divided into four sections. The first two sections discuss the terms involving the partial derivatives of the aerodynamic and structural equations respectively. The last two sections cover the partial derivatives of the drag coefficient (C_D) and the KS function. These four terms are differentiated with respect to the state vectors (w_i, u_j) and the vector of design variables (x_n).

4.3.1 Partial Derivatives of the Aerodynamic Governing Equations

A number of publications describe in detail the terms involved in the aerodynamic adjoint equation and its associated boundary conditions [30, 33, 60, 55]. Although we do not describe these terms in such detail, we do explain the meaning of all the terms, specially those that arise from the inclusion of structural deformations.

The aerodynamic adjoint equation can be written as

$$\frac{\partial \mathcal{A}_k}{\partial w_i}\psi_k = -\frac{\partial I}{\partial w_i}. \tag{4.18}$$

The components of the right-hand side vector are usually non-zero only for those points on the CFD surface mesh. The aerodynamic adjoint used in our work is based on a *continuous* formulation that is derived from the partial differential equations that govern the flow. The adjoint partial differential equations are then discretized using the same scheme and mesh as the flow equations. This is in contrast with the *discrete* approach, where the governing equations of the flow are first discretized and an adjoint version of these discrete equations is then constructed by taking the transpose of $\partial\mathcal{A}_k/\partial w_i$.

The Jacobian $\partial\mathcal{A}_k/\partial w_i$ in the adjoint equation (4.18) represents the variation of the residuals for each cell of the CFD mesh due to changes in the flow solution for every cell in the mesh. When a flow variable at a given cell center is perturbed the residuals of that cell and other cells in its vicinity are modified. The extent of the influence of these flow variable perturbations depends on the stencil used in the flow solver: in our case, a single-level halo of cells is affected. Therefore, even though $\partial \mathcal{A}_k / \partial w_i$ is a very large square matrix, it is also extremely sparse and its non-zero terms can be easily calculated using finite differences. In our coupled-adjoint solver this matrix is never stored explicitly and the adjoint equation (4.15) is solved iteratively, much like the flow solver.

The off-diagonal term $\partial \mathcal{A}_k / \partial u_j$ in the LCA equation (4.16) represents the effect that the structural displacements have on the residuals of the CFD equations through the perturbation of the CFD mesh. When a given structural node moves, both the surface and the interior of the CFD grid must be perturbed, thus affecting a large number of CFD mesh points. Even though the flow variables are constant in the calculation of this partial derivative, the change in the mesh geometry affects the sum of the fluxes, whose variation is easily obtained by recalculating the residuals for the perturbed cells. Because the actual term we want to compute in equation (4.16) is the product of this matrix, $\partial \mathcal{A}_k / \partial u_j$, with the lagged aerodynamic adjoint vector, $\tilde{\psi}_k$, it is possible to multiply each column j by the adjoint vector as it is calculated. This approach eliminates the need to store the complete matrix, since we only need to store a vector with the same dimension that of the adjoint vector.

The term $\partial \mathcal{A}_k / \partial x_n$ in the total sensitivity equation (4.17) represents the direct effect of the design variables on the CFD residuals of all cells in the mesh. For finite-element thickness design variables, this term is identically zero, since these design variables do not affect the CFD residuals explicitly. For shape design variables, this Jacobian is similar to $\partial \mathcal{A}_k / \partial u_j$, since a change in an OML design variable also perturbs the CFD grid. In the present work, this term is calculated by finite differencing, since the mesh perturbation algorithm is very efficient. Again, the term we ultimately want is the vector that results from the product $\psi_k \, \partial \mathcal{A}_k / \partial x_n$, and the matrix multiplication can be performed as each row of the matrix is calculated.

The number of CFD mesh perturbations required for the calculation of the aerodynamic equation sensitivities is equal to $N_x + N_{\mathcal{S}} M$, where N_x is the number of design variables, $N_{\mathcal{S}}$ is the number of surface degrees of freedom of the structural model and M is the number of times the adjoint vectors are exchanged in the iterative solution of the LCA equations (4.15,4.16). The cost of computing $\psi_k \, \partial \mathcal{A}_k / \partial x_n$ is proportional to N_x, while the computation of $\partial \mathcal{A}_k / \partial u_j \, \tilde{\psi}_k$ is proportional to $N_{\mathcal{S}}$ and must be performed M times. Since $N_x + N_{\mathcal{S}} M$ can be very large, it is extremely important that the mesh perturbation procedure be efficient. This is achieved in our case because the mesh perturbation algorithm is completely algebraic. The fact that the CFD mesh is structured makes it possible to attenuate perturbations applied to the surface throughout the volume mesh. In Section 4.5 we compare the cost of the complete sensitivity analysis to the cost of the mesh perturbations.

4.3.2 Partial Derivatives of the Structural Governing Equations

When using linear finite-element models for structural analysis, the discretized governing equations are given by

$$\mathcal{S}_l = K_{lj} u_j - f_l = 0, \tag{4.19}$$

where, K_{lj} is the global stiffness matrix of the structure, u_j is the vector of nodal displacements, and f_l is the vector of applied nodal forces. In our case, the structural model has a relatively small number of degrees of freedom ($\mathcal{O}(10^3)$), and a Cholesky factorization is appropriate to solve for the unknown displacements. The factorization is explicitly stored and is used to solve the structural equations multiple times with different load vectors. For large finite-element models, where the number of degrees of freedom exceeds $\mathcal{O}(10^5)$, alternative approaches to solving the structural equations (4.19) would be more realistic.

To calculate structural sensitivities using the adjoint method we need the partial derivative of the structural governing equations (4.19) with respect to the displacements, which is nothing more than the global stiffness matrix, i.e.,

$$\frac{\partial \mathcal{S}_l}{\partial u_j} = K_{lj}. \tag{4.20}$$

Hence, the adjoint equations for the structural system are

$$K_{jl} \phi_l = -\frac{\partial I}{\partial u_j}. \tag{4.21}$$

Since the stiffness matrix is symmetric ($K_{lj} = K_{jl}$), the structural adjoint equations (4.21) have the same stiffness matrix as the structural governing equations (4.19), i.e., the system is *self adjoint*. The only difference between these two sets of equations is the vector in right-hand side: instead of the load vector in the governing equations (4.19), the adjoint equations (4.21) have a vector (often referred to as *pseudo load*) related to the function of interest, I. As previously mentioned, the stiffness matrix is factorized once when solving for the displacements and it is therefore possible to reuse this factorization to solve for ϕ_l with only a small additional cost.

The derivative of the structural governing equations with respect to the flow variables, $\partial \mathcal{S}_l / \partial w_i$, is the other off-diagonal term in the aero-structural adjoint equations (4.12). In the LCA equation (4.16) we need this term to compute the lagged term $\partial \mathcal{S}_l / \partial w_i \, \tilde{\phi}_l$. The only term in the governing equations (4.19) that the flow variables affect directly is the applied force, and thus

$$\frac{\partial \mathcal{S}_l}{\partial w_i} = -\frac{\partial f_l}{\partial w_i} = -\frac{\partial f_l}{\partial p_{i'}} \frac{\partial p_{i'}}{\partial w_i}, \tag{4.22}$$

where we note that the flow variables affect the structural forces via the surface pressures, $p_{i'}$. Although the matrix $\partial p_{i'} / \partial w_i$ is rather large, it is very sparse since the surface pressures depend only on a small subset of the flow variables. The matrix $\partial f_l / \partial p_{i'}$ is calculated analytically by examining the procedure that integrates the pressures in the CFD mesh and transfers them to the structural nodes to obtain the applied forces. The resulting matrix, $\partial \mathcal{S}_l / \partial w_i$, is rather large, of $\mathcal{O}(10^3 \times 10^6)$, but is never stored explicitly. Since the term we want is actually the vector $\partial \mathcal{S}_l / \partial w_i \, \tilde{\phi}_l$, we calculate one row at a time and perform the dot product with the structural adjoint vector.

Finally, we also need the partial derivative with respect to the design variables, $\partial \mathcal{S}_l / \partial x_n$. The shape design variables have a direct effect on both the stiffness matrix and the load vector. Although this partial derivative assumes a constant surface pressure field, a variation in the OML affects the transfer of these pressures to structural loads. Hence,

$$\frac{\partial \mathcal{S}_l}{\partial x_n} = \frac{\partial K_{lj}}{\partial x_n} u_j - \frac{\partial f_l}{\partial x_n}. \tag{4.23}$$

The element thickness design variables also affect the stiffness matrix, but not the force and therefore $\partial f / \partial x_n = 0$ in this case. As in the case of $\partial \mathcal{S}_l / \partial w_i$, this matrix is also computed by finite differences and multiplied by the structural adjoint vector, one row at a time, eliminating unnecessary storage overhead.

4.3.3 Partial Derivatives of the Drag Coefficient

When solving the adjoint equations for $I = C_D$, we need the partial derivative $\partial C_D / \partial w_i$ to calculate the right-hand side of the first aero-structural adjoint equation (4.15). The value of C_D only depends on the flow variables corresponding to those cells that lie on the surface of the aircraft, so this vector is very sparse. The non-zero sensitivities in this vector are obtained analytically by differentiating the numerical integration procedure of the surface pressures that produces C_D.

The second aero-structural adjoint equation (4.16) contains another partial derivative of C_D, but this one is taken with respect to the structural displacements. The vector $\partial C_D / \partial u_j$ represents the change in the drag coefficient due to the displacement of the wing while keeping the pressure field constant. The structural displacements affect the drag directly, since they change the wing surface over which the pressure is integrated. This vector of sensitivities is efficiently computed by finite differencing.

Finally, in order to calculate the total sensitivity of the drag coefficient using equation (4.17), we need the term $\partial C_D/\partial x_n$. This represents the change in the drag coefficient due to design variable perturbations, while keeping the pressure and displacement fields constant. In the case of shape perturbations, $\partial C_D/\partial x_n$ is analogous to $\partial C_D/\partial u_j$ because these design variables change the surface of integration. This vector is also inexpensively calculated using finite differences. For structural design variables this term is zero because they do not affect the OML directly.

4.3.4 Partial Derivatives of the KS Function

As discussed in Section 1.4.3, the other set of sensitivities we are ultimately interested in is that of the KS function (1.9), i.e., when $I = \text{KS}$. Since this function depends directly on the stresses we use the chain rule to write,

$$\frac{\partial \text{KS}}{\partial u_j} = \frac{\partial \text{KS}}{\partial g_m} \frac{\partial g_m}{\partial \sigma_m} \frac{\partial \sigma_m}{\partial u_j}. \tag{4.24}$$

Differentiating the KS function (1.9) definition we can write the first term as

$$\frac{\partial \text{KS}}{\partial g_m} = \left[\sum_{m'} \mathrm{e}^{-\rho g_{m'}} \right]^{-1} \mathrm{e}^{-\rho g_m} . \tag{4.25}$$

The second term is easily derived from the definition of the stress constraints (1.8),

$$\frac{\partial g_m}{\partial \sigma_m} = -\frac{1}{\sigma_y}. \tag{4.26}$$

To obtain the third term of equation (4.24) we consider the expression that relates the stresses to the displacement field,

$$\sigma_m = S_{mj} u_j. \tag{4.27}$$

Given the linear nature of this relationship, the partial derivative we need is simply

$$\frac{\partial \sigma_m}{\partial u_j} = S_{mj}. \tag{4.28}$$

Using these results we can rewrite the partial derivative (4.24) as

$$\frac{\partial \mathrm{KS}}{\partial u_j} = -\left[\sigma_y \sum_{m'} \mathrm{e}^{-\rho g_{m'}}\right]^{-1} \mathrm{e}^{-\rho g_m} S_{mj}. \tag{4.29}$$

We use this term in the right-hand side of the structural adjoint equation (4.21) — or equation (4.15) in the aero-structural case — the to solve for the adjoint vector that corresponds to the sensitivities of the KS function.

For the case where $I = \mathrm{KS}$, the right-hand-side of the aerodynamic adjoint equation (4.15) includes $\partial\mathrm{KS}/\partial w_i$. This term is zero, since the stresses do not depend explicitly on the loads. They only depend on the loads implicitly, through the displacements.

Finally, the last partial derivative of the KS function, $\partial\mathrm{KS}/\partial x_n$, appears in the total sensitivity equation (4.17). This term represents the variation of the lumped stresses for fixed loads and displacements. As in the case of the partial derivative with respect to the displacements (4.24), we can use the chain rule to write

$$\frac{\partial \mathrm{KS}}{\partial x_n} = \frac{\partial \mathrm{KS}}{\partial g_m}\frac{\partial g_m}{\partial \sigma_m}\frac{\partial \sigma_m}{\partial x_n}. \tag{4.30}$$

Since we have derived the two first partial derivative terms (4.25,4.26) we are left with only one new term, the partial derivative of the stresses with respect to the design variables. Taking the derivative of the stress-displacement relationship (4.27) yields

$$\frac{\partial \sigma_m}{\partial x_n} = \frac{\partial S_{mj}}{\partial x_n} u_j, \tag{4.31}$$

where $\partial S_{mj}/\partial x_n$ is calculated using finite differences. For element thickness design variables, $\partial\sigma_m/\partial x_n = 0$. However, when the OML is perturbed, the stresses in a given element can vary if its shape is distorted.

4.4 Sensitivity Validation Results

We now present the results of the implementation of the LCA method to the aero-structural analysis framework described in Chapter 2. The results shown herein correspond to the supersonic business jet configuration detailed in the next chapter, in Section 5.1.

To compute the flow for this configuration, we use the CFD mesh shown in Figure 4-3. This is a multiblock Euler mesh with 36 blocks and a total of 220,000 mesh points. The structural model of the wing is also shown in Figure 4-3, and a more detailed drawing of

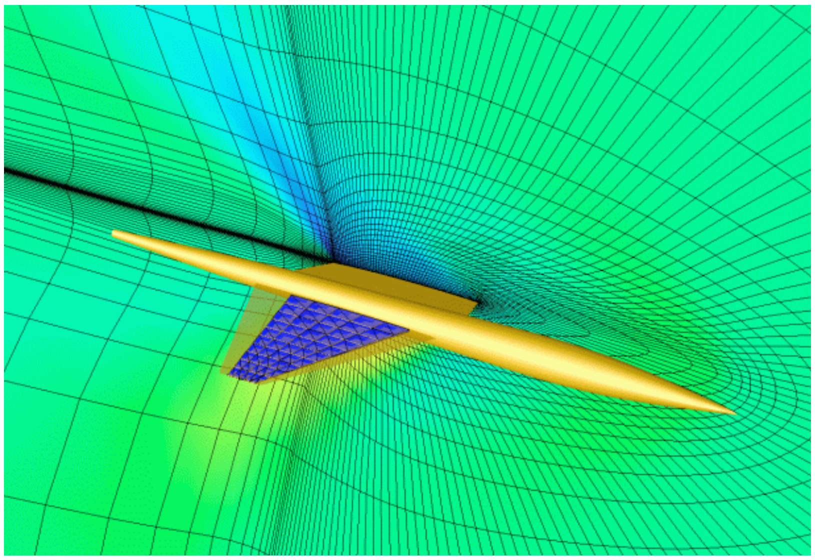

Figure 4-3: Aero-structural model of the supersonic business jet configuration, showing a slice of the flow mesh and the internal structure of the wing.

the wing model can be found in Figure 2-1. The model consists of a wing box with six spars evenly distributed from 15% to 80% of the chord. Ribs are distributed along the span at every tenth of the semispan. A total of 640 finite elements were used in the construction of this model.

To gain confidence in the effectiveness of the aero-structural coupled-adjoint sensitivities for use in design optimization, we must ensure that the values of the gradients are accurate. For this purpose, we chose to validate the four sets of sensitivities discussed below. For comparison purposes only, we compute the exact discrete value of these sensitivities using the complex-step derivative approximation presented in Chapter 3.

In this sensitivity study two different functions are considered: the aircraft drag coefficient, C_D, and the KS function (1.9). The sensitivities of these two quantities with respect to both OML shape design variables and structural design variables are computed and discussed. The design variables are as described in Section 1.4.3.

4.4.1 Drag Coefficient Sensitivities

The values of the aero-structural sensitivities of the drag coefficient with respect to shape perturbations are shown in Figure 4-4. The ten shape perturbations were chosen to be Hicks–Henne functions distributed chordwise on the upper surface of two adjacent airfoils around the quarter span. The plot shows very good agreement between the coupled-adjoint

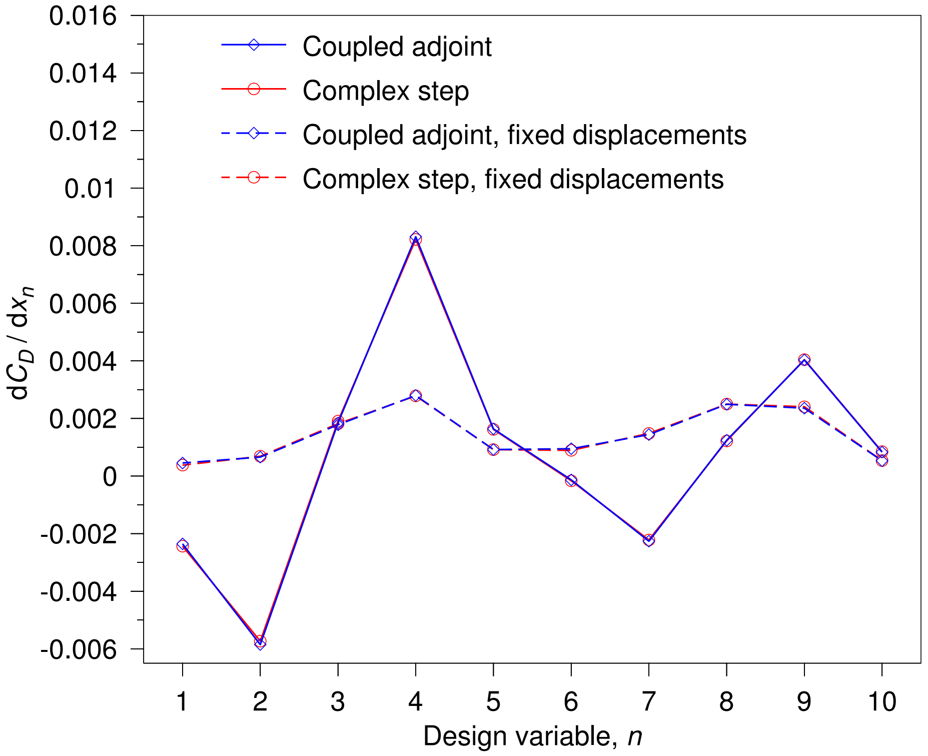

Figure 4-4: Sensitivities of the drag coefficient with respect to shape perturbations.

and the complex-step results, with an average relative error between the two of only 3.5%. Note that all these sensitivities are *total* sensitivities in the sense that they account for the coupling between aerodynamics and structures.

To verify the need for taking the coupling into account, the same set of sensitivities was calculated for fixed structural displacements, where the displacement field is frozen after the aero-structural solution. This is, in some sense, similar to assuming that the wing, after the initial aeroelastic deformation, is infinitely rigid as far as the computation of sensitivities is concerned. When calculating these sensitivities using the complex step, the reference solution is aero-structural, but only the flow solver is called for each shape perturbation. When using the adjoint method, this is equivalent to solving only the aerodynamic adjoint in (4.15) and omitting the partial derivatives of $\mathcal{S}_l$ in the gradient calculation (4.17). Figure 4-4 shows that the single-system sensitivities exhibit significantly lower magnitudes and even opposite signs for many of the design variables when compared with the coupled sensitivities. The use of single-discipline sensitivities would therefore lead to erroneous design decisions.

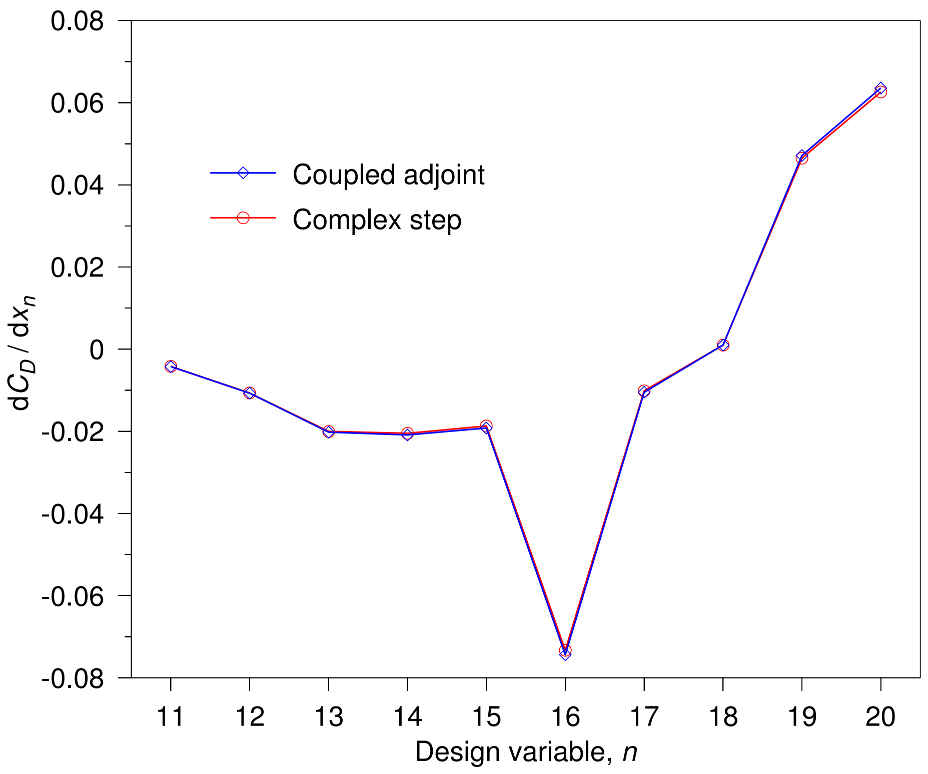

Figure 4-5: Sensitivities of the drag coefficient with respect to structural thicknesses.

Figure 4-5 also shows the sensitivity of the drag coefficient, this time with respect to the thicknesses of five skin groups and five spar groups distributed along the span. The agreement in this case is even better; the average relative error is only 1.6% respectively. Even though these are sensitivities with respect to internal structural variables that do not modify the jig OML, coupled sensitivity analysis is still required.

4.4.2 KS Function Sensitivities

The sensitivities of the KS function with respect to the two sets of design variables described above are shown in Figures 4-6 and 4-7. The results show that the coupled-adjoint sensitivities are extremely accurate, with average relative errors of 2.9% and 1.6%. In Figure 4-7 we observe that the sensitivity of the KS function with respect to the first structural thickness is much higher than the remaining sensitivities. This markedly different magnitude is due to the fact that this particular structural design variable corresponds to the thickness of the top and bottom skins of the wing bay closest to the root, where the stress is the highest at this particular load condition.

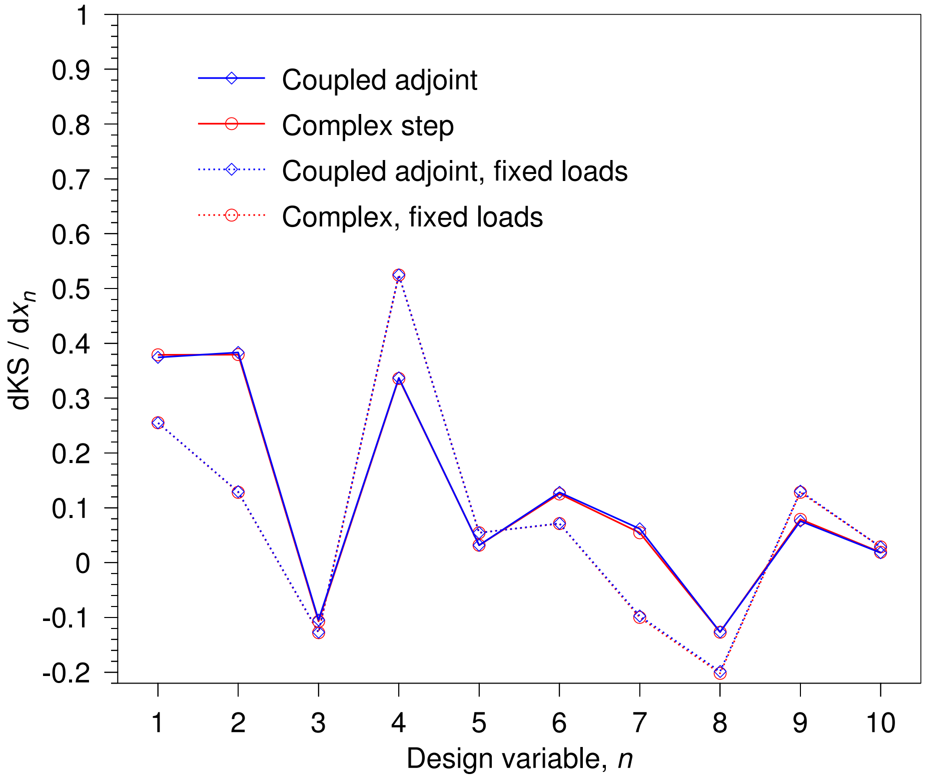

Figure 4-6: Sensitivities of the KS function with respect to shape perturbations.

The sensitivities of the KS function for fixed loads are also shown in Figures 4-6 and 4-7. Using the complex-step method, these sensitivities were calculated by calling only the structural solver after the initial aero-structural solution, which is equivalent to using just equations (4.16, 4.17) without the partial derivatives of $\mathcal{A}_k$ after solving the aero-structural system. The difference in these sensitivities when compared to the coupled ones is not as dramatic as in the fixed displacements case shown in Figure 4-4, but it is still significant.

4.5 Computational Efficiency Study

4.5.1 Comparison of the Coupled Adjoint with Finite Differencing

The cost of calculating a gradient vector using either the finite-difference or the complex-step methods is expected to be linearly dependent on the number of design variables. This expectation is confirmed in Figure 4-8 where the gradient calculation times are shown for increasing numbers of design variables. The time axis is normalized with respect to the

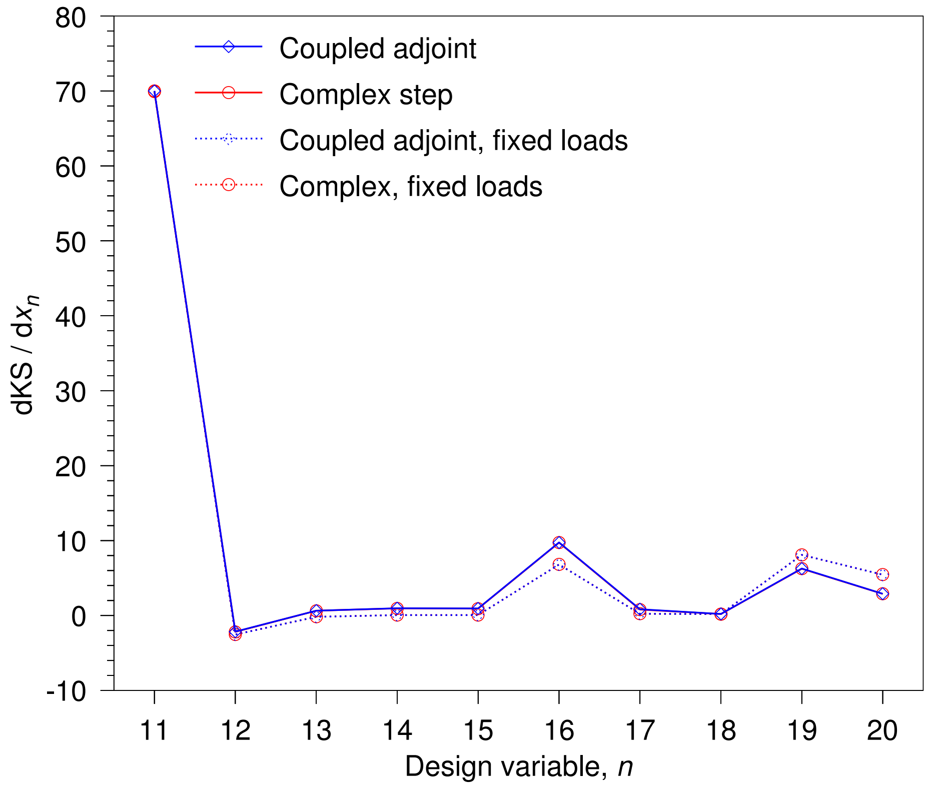

Figure 4-7: Sensitivities of the KS function with respect to structural thicknesses.

time required for a single aero-structural solution (98 seconds on 9 processors of an SGI Origin 2000).

The cost of a finite-difference gradient evaluation can be linearly approximated by the equation $1.0 + 0.38 \times N_x$, where N_x is the number of design variables. Notice that one might expect this method to incur a computational cost equivalent to one aero-structural solution per additional design variable. The cost per additional design variable is lower than this because each additional aero-structural calculation does not start from a uniform flow-field initial condition, but from the previously converged solution, which is closer to the final solution.

The same applies to the cost of the complex-step method. Because the function evaluations require complex arithmetic, the cost of the complex step method is, on average, 2.4 times higher than that of finite differencing. However, this cost penalty is worthwhile since there is no need to find an acceptable step size *a priori*, as is the case for finite-difference approximations [46, 47].

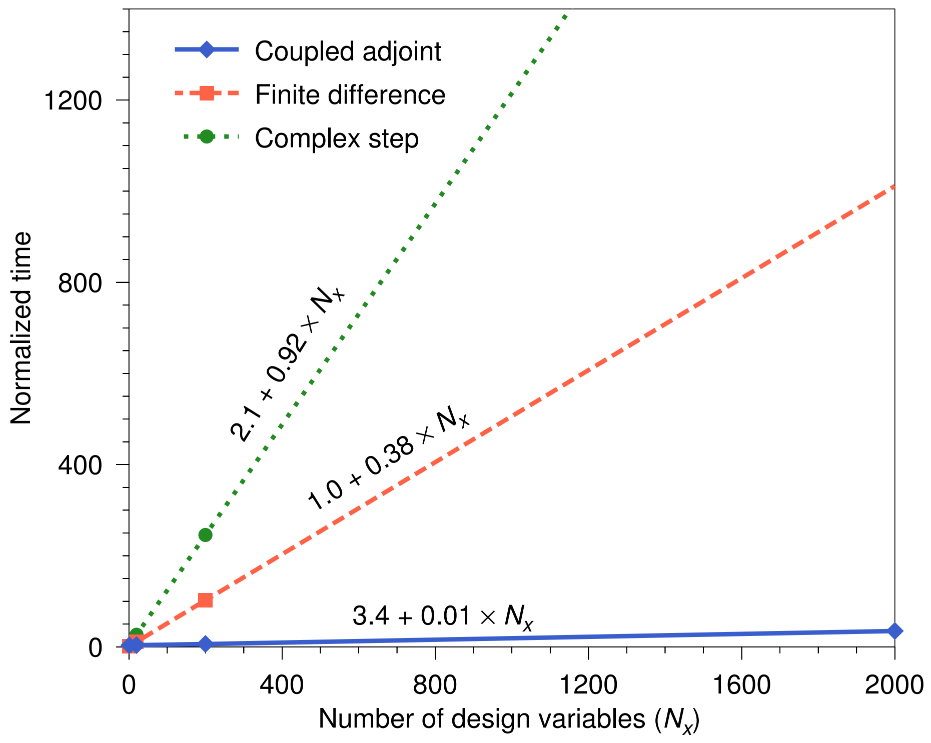

Figure 4-8: Computational time vs. number of design variables for finite differencing, complex step and coupled adjoint. The time is normalized with respect to the time required for one aero-structural solution.

The cost of computing sensitivities using the coupled-adjoint procedure is in theory independent of the number of variables. Using our implementation, however, some of the partial derivatives in the total sensitivity equation (4.17) are calculated using finite differences and therefore, there is a small dependence on the number of variables. The line representing the cost of the coupled adjoint in Figure 4-8 has a slope of 0.01 which is between one and two orders of magnitude less than the slope for the other two lines.

In short, the cost of computing sensitivities with respect to hundreds or even thousands of variables is acceptable when using the coupled-adjoint approach, while it is impractical to use finite-differences or the complex-step method for such a large number of design variables, even with current state-of-the-art parallel computing systems.

4.5.2 Coupled-Adjoint Solution

The constant terms in the equations for the straight lines of Figure 4-8 represent the cost of each procedure when no sensitivities are required. For the finite-difference case, this is

Aero-structural solution	1.000
Aero-structural adjoint	
Aerodynamic adjoint equation (4.15)	0.597
RHS of equation (4.15)	0.642
Structural adjoint equation (4.16)	< 0.001
RHS of equation (4.16)	1.203

Table 4.1: Computational times for solving the LCA equations (4.15,4.16).

equivalent to one aero-structural solution, and hence the constant is 1.0. When performing the aero-structural solution using complex arithmetic, the cost rises to 2.1 times the real arithmetic solution.

The cost of computing the coupled-adjoint vectors (without computing the gradients) is 3.4. This cost includes the aero-structural solution, which is necessary before solving the adjoint equations, and hence the aero-structural adjoint computation alone incurs a cost of 2.4. To gain a better understanding of how this cost it divided, we timed the computation for four different components of the aero-structural adjoint equations (4.15, 4.16) as shown in Table 4.1.

The cost of solving the lagged aerodynamic adjoint equation (4.15) — not including the computation of the right-hand-side vector — is about 0.6 times the aero-structural solution. This is expected, since the cost of solving the adjoint equations of a given system is usually similar to the cost of solving the corresponding governing equations. The cost of computing right-hand-side of the same equation, is also about 0.6. This cost is almost exclusively due to the lagged term $\partial \mathcal{S}_l / \partial w_i \, \tilde{\phi}_l$, which is partially computed using finite differences, as explained in Section 4.3.2. Note that the cost of computing this term is proportional to the number of OML points, which is 4,200 in our calculations.

The computation time for solving the lagged structural adjoint equation (4.16) is negligible. Again, this does not account for the computation of the right-hand side of that equation. The cost of solving this equation is so low because the factorization of the matrix has already been computed and only a back-solve operation is required. The computation time for the right-hand side of this equation, however, is rather high: 1.2. Again, this is almost solely due to the lagged term, which is $\partial \mathcal{A}_k / \partial u_j \, \tilde{\psi}_k$ in this case. As explained in

Section 4.3.1 this term is computed using finite differences and therefore its cost is proportional to the number of structural surface degrees of freedom, which in this case is equal to 396.

In Chapter 5 we will see that in realistic aero-structural design problems with hundreds or even thousands of design variables, there is a considerable reduction in computational cost when using the coupled-adjoint method as opposed to either finite differences or the complex step. This is due to the fact that the cost associated with the adjoint method is practically independent of the number of design variables.

Chapter 5

Aero-Structural Optimization

This chapter presents the results of the aero-structural optimization of a supersonic business jet configuration using the coupled adjoint method detailed in Chapter 4, in conjunction with a gradient-based optimizer.

5.1 Supersonic Business Jet Design Case

The aircraft configuration used in this example is the supersonic business jet shown in Figure 5-1. This configuration is being developed by the ASSET Research Corporation and is designed to achieve a high percentage of laminar flow on the low-sweep wing [67, 36]. The greater the extent of laminar flow over the wing, the lower the aerodynamic friction drag, resulting in decreased fuel consumption. The main objective of this concept is to make supersonic flight more economical. The aircraft is to fly at Mach 1.5 and have a range of 5,300 nautical miles. Additional specifications are listed in Table 5.1.

Detailed mission analysis for this aircraft has determined that one count of drag ($\Delta C_D = 0.0001$) is worth 310 pounds of empty weight. A very similar result can be obtained from the Breguet range equation

$$R = \frac{V}{c}\frac{C_L}{C_D}\ln\frac{W_i}{W_f}, \tag{5.1}$$

where V is the cruise velocity and c is the thrust specific fuel consumption of the powerplant. C_L/C_D is the ratio of lift to drag, and W_i/W_f is the ratio of initial and final cruise weights of the aircraft. The final cruise weight is the initial cruise weight minus the weight of the fuel used in the cruise segment.

If we assume fixed initial cruise weight, velocity and lift coefficient, and chose to minimize the final cruise weight, the only two quantities that vary in the Breguet equation are W_f

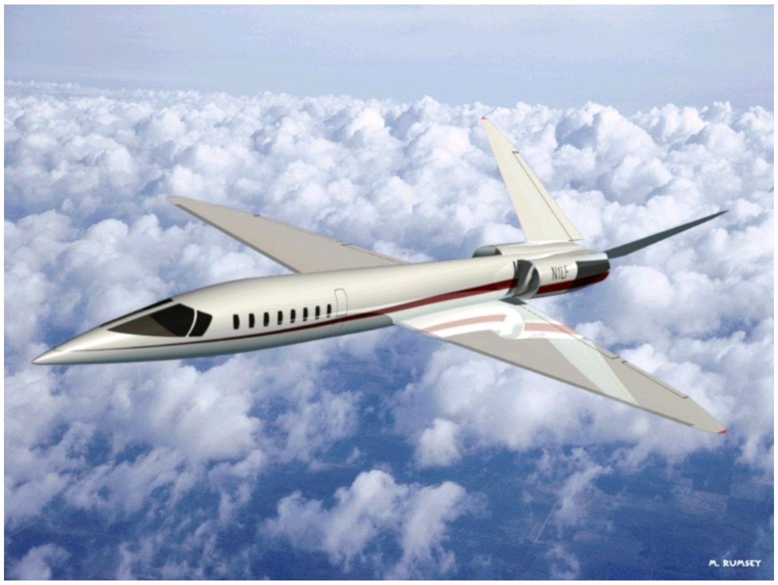

Figure 5-1: Natural laminar flow supersonic business jet configuration.

and C_D. Linearizing the Breguet range equation with respect these two variables yields

$$R = -\alpha C_D - \beta W_f, \tag{5.2}$$

where

$$-\alpha = \frac{\partial R}{\partial C_D} = -\frac{V}{c}\frac{C_L}{C_D^2}\ln\frac{W_i}{W_f}, \tag{5.3}$$

and

$$-\beta = \frac{\partial R}{\partial W_f} = -\frac{V}{c}\frac{C_L}{C_D}\frac{1}{W_f}. \tag{5.4}$$

The ratio of these two partial derivatives α and β represents the relative worth of the drag and weight and we write this ratio as

$$\frac{\alpha}{\beta} = \frac{W_f}{C_D}\ln\frac{W_i}{W_f}. \tag{5.5}$$

For this particular design, using the data from Table 5.1, yields a ratio of $\alpha/\beta = 3.04 \times 10^6$. This is very close to the ratio given by the much more detailed mission analysis that was previously mentioned, for which $\alpha/\beta = 310/0.0001 = 3.10 \times 10^6$.

We can now maximize the linearized range of the business jet configuration by minimizing the objective function

$$I = \alpha C_D + \beta W, \tag{5.6}$$

Performance		
Cruise Mach number	1.5	
Range	5,300	nm
Take-off gross weight (TOGW)	100,000	lbs
Zero-fuel weight (ZFW)	47,500	lbs
Cruise altitude	51,000	ft
Cruise lift coefficient	0.1	
Cruise drag coefficient	0.0116	
Cruise TSFC	0.86	

Wing geometry		
Reference area	1750	ft^2
Aspect ratio	3.0	
Taper ratio	0.218	

Table 5.1: Specifications for the natural laminar flow supersonic business jet.

where C_D is the drag coefficient, W is the wing structural weight in pounds and $\alpha/\beta = 3.1 \times 10^6$. Note that we are able to substitute the final cruise weight by the wing structural weight, since the difference between the two is constant.

The optimization problem we ultimately want to solve is the simplified aircraft design problem discussed in Chapter 1, i.e.,

$$\begin{aligned} \underset{x_n \in \mathbb{R}^n}{\text{minimize}} \quad & I = \alpha C_D + \beta W \\ \text{subject to} \quad & C_L = C_{L_T} \\ & \text{KS} \geq 0 \\ & x_n \geq x_{n_{\min}}. \end{aligned} \tag{5.7}$$

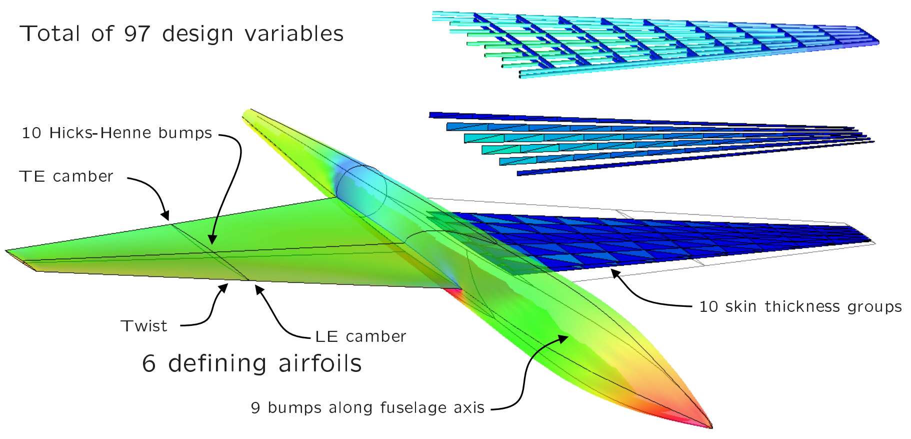

Figure 5-2: Description of the OML and structural design variables.

In this particular case, we consider only one cruise condition for the calculation of the drag coefficient, and a separate maneuver condition for the structural stress constraints. The cruise condition has a target lift coefficient of 0.1 while the one for the maneuver condition is 0.2.

The geometry of the supersonic business jet was simplified for the purposes of solving this aircraft optimization problem. Again, this represents only a proof-of-concept case to demonstrate this design methodology and is not meant to represent a realistic full-configuration case. The baseline geometry is the wing-body configuration shown in Figure 5-8. The body is axisymmetric and exhibits a smooth variation in the cross-sectional distribution in the longitudinal axis, with maximum area at around one third of the body length from the nose. The wing sections are biconvex and have a constant thickness-to-chord ratio of 2%.

A description of all the design variables that are used in this optimization problem is shown in Figure 5-2. In order to parameterize the shape of the aircraft, we have chosen sets of design variables that apply to both the wing and the fuselage. The wing shape is modified by optimization at six defining stations uniformly distributed from the wing-body intersection to the tip of the wing. The shape modifications of these defining stations are linearly lofted to a zero value at the previous and next defining stations. On each defining station, the twist, the leading and trailing edge camber, and five Hicks–Henne bump functions on both the upper and lower surfaces are allowed to vary. The leading and trailing edge camber modifications are not applied at the first defining station. This yields a total of 76 OML design variables on the wing. The wing planform remains constant in this design problem, since planform optimization is only meaningful if additional disciplines and constraints are taken into account.

The shape of the fuselage is parameterized in such a way that its camber is allowed to vary while the total volume remains constant. This is accomplished with 9 bump functions evenly distributed in the streamwise direction starting at the 10% fuselage station. Fuselage nose and trailing edge camber functions are added to the fuselage camber distribution in a similar way to what was done with the wing sections.

The structural sizing is accomplished with 10 design variables, which correspond to the skin thicknesses of the top and bottom surfaces of the wing. Each group is formed by the plate elements located between two adjacent ribs. All structural design variables are constrained to exceed a specified minimum gauge value.

The configuration is therefore parameterized with a total of 97 design variables. As mentioned in an earlier section, the cost of aero-structural gradient information using our coupled-adjoint method is effectively independent of the number of design variables: in more realistic full configuration test cases that we are about to tackle, 500 or more design variables will be necessary to describe the shape variations of the configuration (including nacelles, diverters, and tail surfaces) and the sizing of the structure.

5.2 Integrated Aero-Structural Optimization

All the sensitivities required to solve this problem using gradient-based optimization are computed using the LCA method described in Chapter 4. The optimization process is carried out using the nonlinear constrained optimizer NPSOL [20]. Euler calculations are performed on a wing-body 36-block mesh that is constructed from the decomposition of a $193 \times 33 \times 49$ C-H mesh. During the process of the optimization, all flow evaluations are converged to 5.3 orders of magnitude of the average density residual and the C_L constraint is achieved to within 10^{-6}.

The optimization history of this design case is shown in Figure 5-3. The figure shows the values of the coefficient of drag (in counts), the wing structural weight (in lbs), and the value of the KS function for successive major design iterations. Note that the structural constraints are satisfied when the KS function is positive. Because of the approximate nature of the KS function, all structural constraints may actually be satisfied for small but negative values of the KS function.

The values for drag coefficient, wing weight and KS function for the baseline design are shown in the leftmost points of Figure 5-3 (zeroth major iteration). The KS function is positive (KS = 0.115) indicating that all stress constraints are satisfied at the maneuver

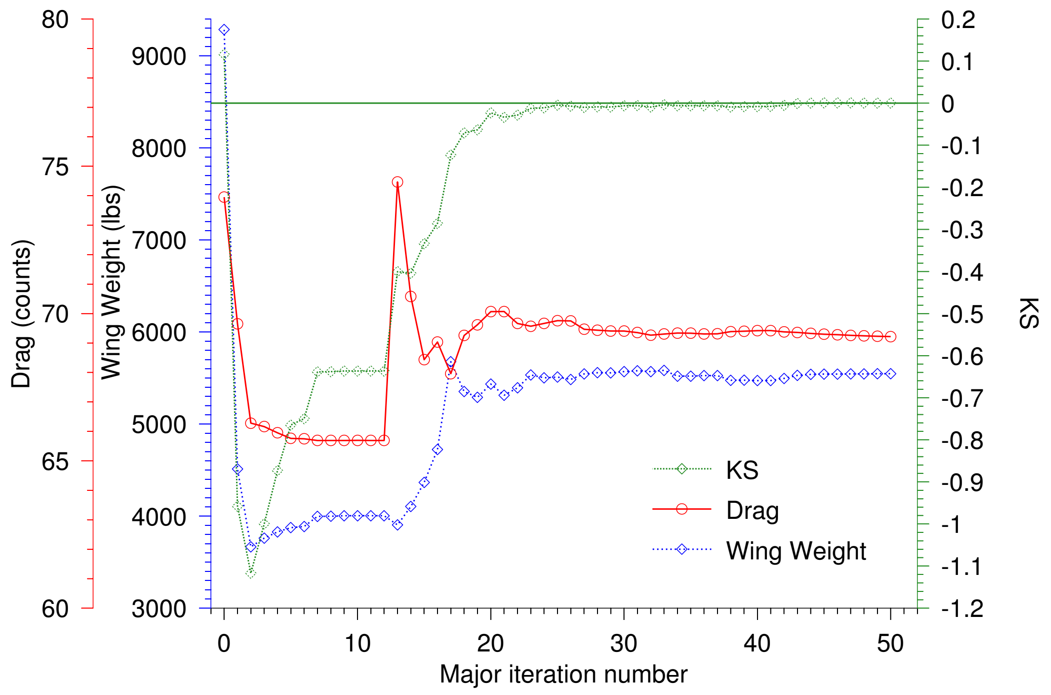

Figure 5-3: Convergence history of the aero-structural optimization.

condition. The cruise drag coefficient of 74.04 counts and a structural weight of 9,285 lbs. The surface density distribution for the cruise condition and the structural stresses for this baseline design are shown in Figure 5-8. Note that the drag coefficients quoted herein do not include either the drag resulting from the empennage and nacelles, or the friction drag.

In the first two design iterations, the optimizer takes large steps in the design space, resulting in a drastic reduction in both C_D and W. However, this leads in a highly infeasible design that exhibits maximum stresses that are up to 2.1 times the yield stress of the material. After these initial large steps, the optimizer manages to decrease the norm of the constraint violation. This seems to have been accomplished by increasing the structural skin thicknesses, since the weight increases while the drag is further reduced. Towards major iteration 10, there is no visible progress for several iterations while the design remains infeasible. A large step is taken in iteration 13 that results in a sudden increase in feasibility accompanied by an equally sudden increase in C_D. The optimizer has established that the only way to obtain a feasible design is by increasing the wing thickness (with the consequent increases in C_D and weight) and the structural thicknesses. From that point on, the optimizer rapidly converges to the optimum. After 43 major iterations, the KS constraint is reduced to $\mathcal{O}(10^{-4})$ and all stress constraints are satisfied. The aero-structurally optimized

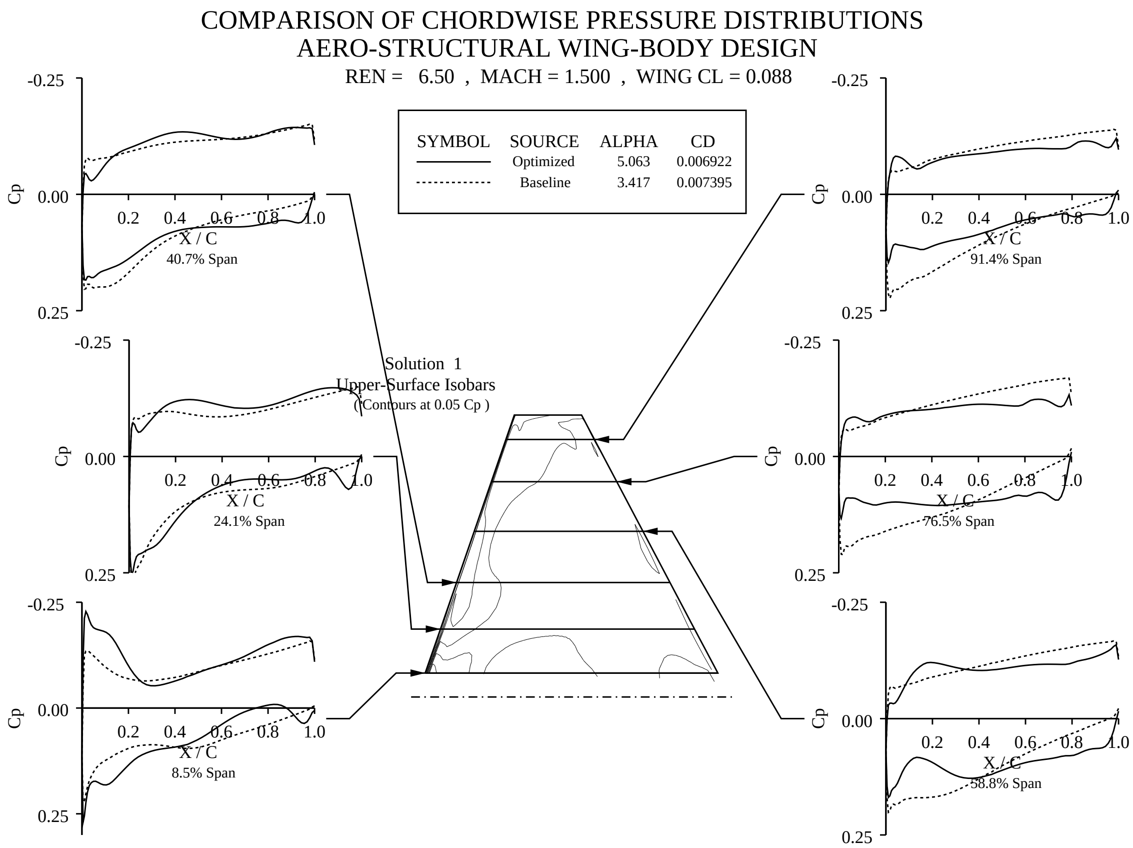

Figure 5-4: Wing pressure coefficient distributions for the baseline and optimized designs.

result has $C_D = 0.006922$ and a total wing structure weight of 5,546 lbs.

The fact that NPSOL is able to find the optimum for a constrained nonlinear problem with almost 100 design variables in less than 50 major design iterations is truly remarkable. NPSOL also showed to be rather robust, as it was able to recover from several iterations of no significant improvement. We should note that NPSOL has now been succeeded by SNOPT [21] and that this new optimizer would offer even better performance.

The visualizations of the optimized configuration is shown in Figure 5-9. Measures of performance and feasibility are written in the first section of Table 5.2. The left half of Figure 5-9 shows the surface flow density distribution at the cruise condition with the corresponding structural deflections at the cruise condition for the design. The right half shows an exploded view of the stress distribution on the structure (spar caps, spar shear webs, and skins, from top to bottom) at the $C_L = 0.2$ maneuver condition.

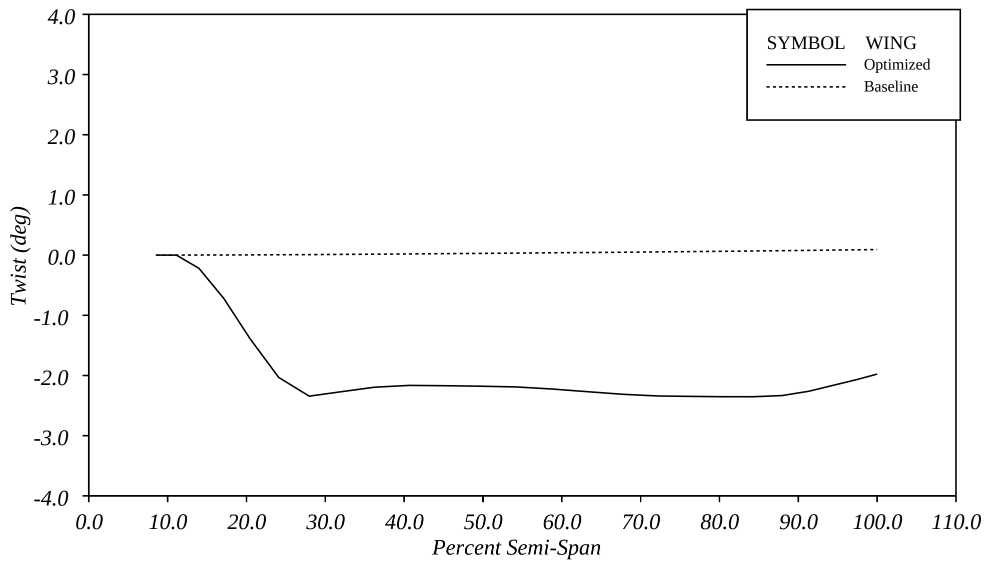

Figure 5-5: Spanwise twist distribution of the baseline and optimized designs.

By comparing the baseline and optimized results of Figures 5-8 and 5-9 one can appreciate that the surface density distributions changed substantially at the cruise condition. A more detailed view of the aerodynamics of both the baseline and optimized configurations is shown in the pressure coefficient distribution of Figure 5-4.

It is also worth noting that about half of the improvement in the C_D of the optimized configuration results from drastic changes in the fuselage shape: both front and aft camber have been added to distribute the lift more evenly in the streamwise direction in order to reduce the total lift-dependent wave drag.

The structural element stresses at the maneuver condition also exhibit dramatic changes. In fact, as expected from a design case with a single load condition, the optimized structure is nearly fully-stressed, except in the outboard sections of the wing, where the minimum gauge constraints are active.

The spanwise twist distribution is more clearly seen in Figure 5-5. the twist distribution has changed dramatically through a combination of both OML design variables and aeroelastic deflections by adding almost 2 degrees of negative twist along most of the span. Since the wing washout angle was constrained to a minimum of -2 degrees, this is an indication of the fact that additional improvements in the induced drag of the configuration may result from further modifications to the twist distribution.

When comparing the wings of Figures 5-8 and 5-9 we can see that the spanwise distributions of airfoil maximum thickness differ significantly. The baseline wing has a linear

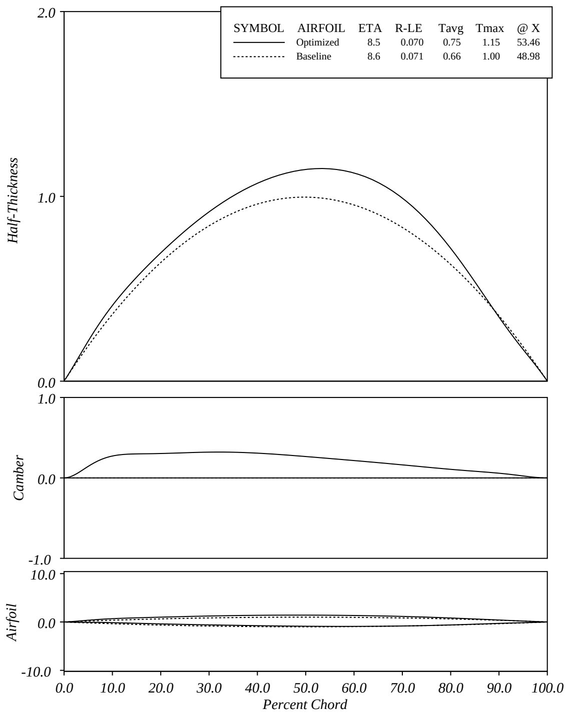

Figure 5-6: Airfoil geometry at the root.

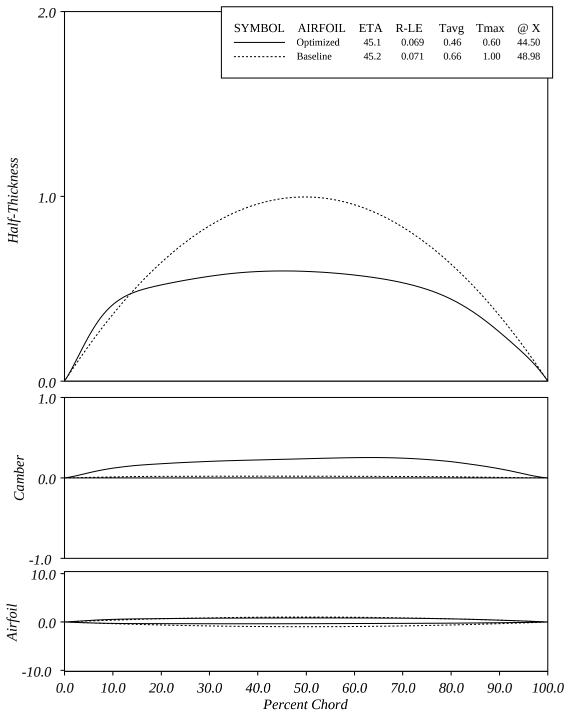

Figure 5-7: Airfoil geometry at mid semi-span.

distribution due to the constant thickness-to-chord ratio. The optimized design has thicker airfoils near the root, where the bending moments are highest, but then quickly decreases the airfoil thickness-to-chord ratio below the baseline values towards the the tip. The airfoil geometries for the root and mid semi-span sections are shown in Figures 5-6 and 5-7, respectively.

In order to solve this optimization problem, total of 50 major design iterations including aero-structural analyses, coupled adjoint solutions, gradient computations, and line searches were performed in approximately 20 hours of wall clock time using 18 processors of an SGI Origin 3000 system (R12000, 400 MHz processors). Since these are not the fastest processors currently available, we expect that much larger models can be optimized with overnight turnaround in the near future.

5.3 Comparison with Sequential Optimization

The usefulness of a coupled aero-structural optimization method can only be measured in comparison with the results that can be obtained using current state-of-the-art practices.

In the case of aero-structural design, the typical approach is to carry out aerodynamic shape optimization with artificial airfoil thickness constraints meant to represent the effect of the structure, followed by structural optimization with a fixed OML. As discussed in Section 1.4.2, sequential optimization cannot be guaranteed to convergence to the true optimum of a coupled system. In order to determine the difference between the optima achieved by fully-coupled and sequential optimizations, we also performed one cycle of sequential optimization within the analysis and design framework.

To prevent the optimizer from thinning the wing to an unreasonable degree during the aerodynamic shape optimization, 5 thickness constraints are added to each of the 6 defining stations for a total of 30 linear constraints. These constraints are such that, at the points where they are applied, the wing box is not allowed to get any thinner than the original design.

After the process of aerodynamic shape optimization is completed, the initial C_D has decreased to 0.006992 as shown in the lower portion of Table 5.2. After fixing the OML, structural optimization is performed using the maneuver loads for the baseline configuration at $C_L = 0.2$. The structural optimization process reduces the weight of the wing structure to 6,567 lbs.

We can now compare the results of the fully coupled optimization in the previous section and the outcome of the process of sequential optimization. The differences are clear: the coupled aero-structural optimization was able to achieve a design with a range of 7,361 nm, which is 224 nm higher than that obtained from the sequential optimization.

Finally, note that since sequential optimization neglects the aero-structural coupling in the computation of maneuver loads, there is no guarantee that the resulting design will be feasible. In fact, the aero-structural analysis shows that the value of the KS function is slightly negative.

	C_D (counts)	KS	$\sigma_{\text{max}}/\sigma_{\text{yield}}$	ZFW (lbs)	Range (nm)
Baseline	73.95	1.15×10^{-1}	0.87	47,500	6,420
Integrated optimization	69.22	-2.68×10^{-4}	0.98	43,761	7,361
Sequential optimization					
Aerodynamic optimization					
Baseline	74.04				
Optimized	69.92				
Structural optimization					
Baseline		1.02×10^{-1}	0.89	47,500	
Optimized		1.45×10^{-8}	0.98	44,782	
Aero-structural analysis	69.92	-9.01×10^{-3}	0.99		7,137

Table 5.2: Comparison between the integrated and sequential approaches to aero-structural optimization.

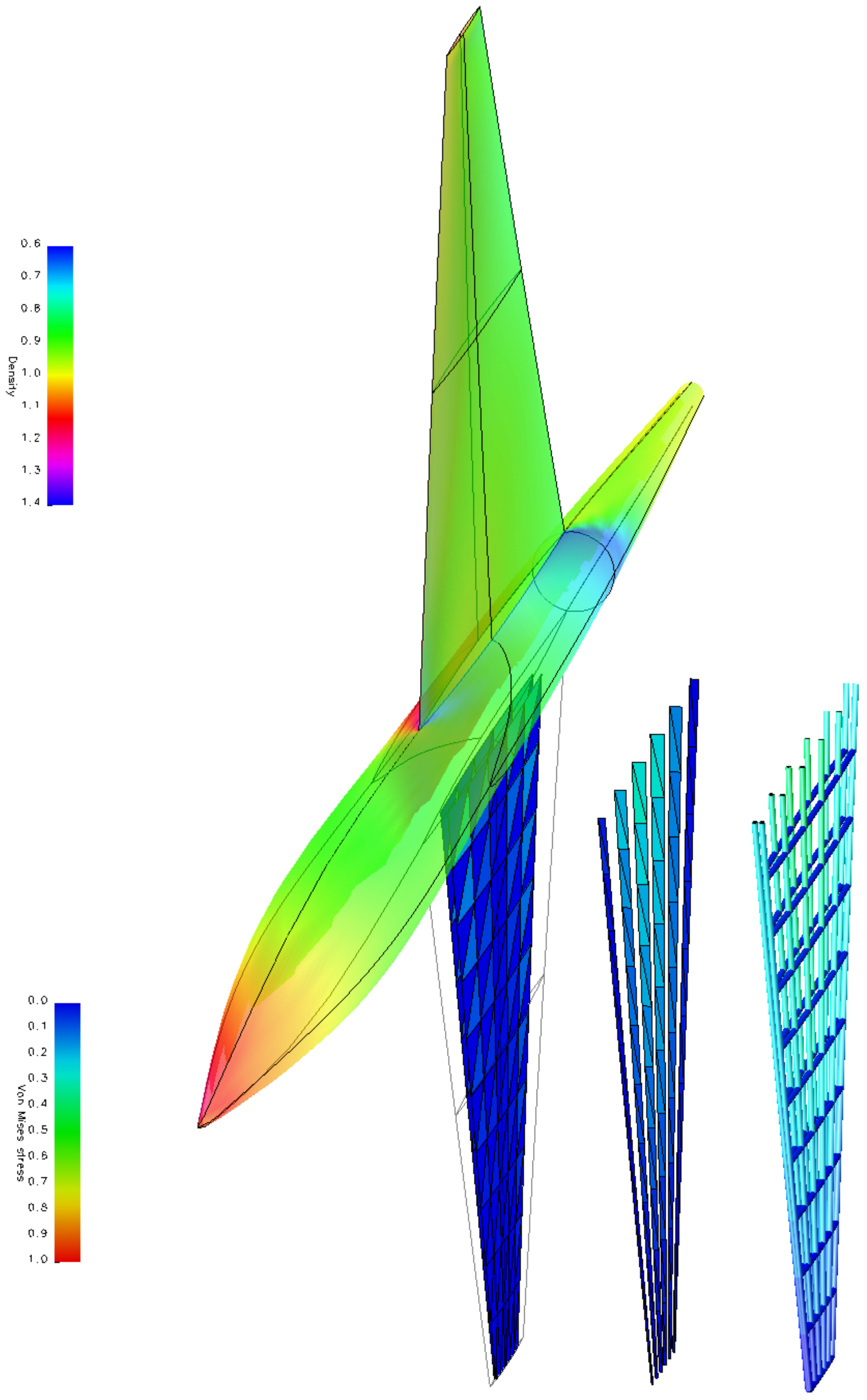

Figure 5-8: Baseline configuration for the supersonic business jet showing surface densities at the cruise condition and structural stresses at the maneuver condition. The density is normalized by the freestream value and the von Mises stresses are normalized by the material yield stress.

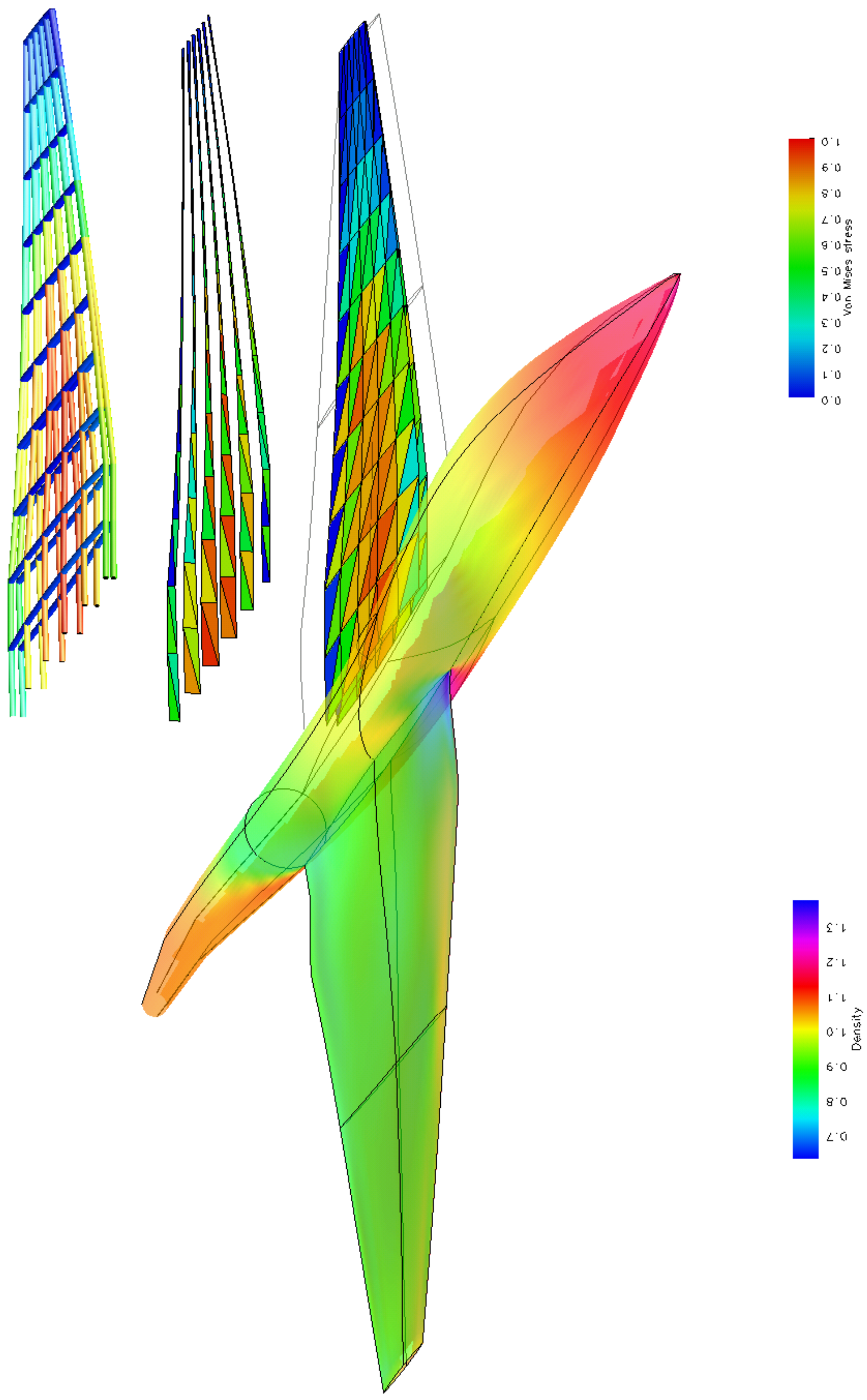

Figure 5-9: Optimized supersonic business jet design.

Chapter 6

Conclusions and Development Directions

The motivation for this dissertation was to introduce high-fidelity analysis tools to the field of multidisciplinary design of aircraft configurations. Two core disciplines were considered: aerodynamics and structures, leading to the development of a framework for high-fidelity aero-structural analysis. In order to reap the benefits of this high-fidelity framework, the design was parameterized using a large number of variables. Since optimization problems with large numbers of variables are solved most efficiently using gradient-based optimization algorithms, it quickly became apparent that the major obstacle to performing high-fidelity aero-structural optimization was the computation of the gradients. The contributions of this dissertation are associated to two methods for the computation of multidisciplinary sensitivities: the aero-structural adjoint method, which fulfilled the premise of high-fidelity aero-structural optimization, and the complex-step derivative approximation, which provided an indispensable benchmark.

An adjoint method for coupled sensitivity analysis of high-fidelity aero-structural systems was presented. The aero-structural adjoint sensitivity equations were compared with other formulations to show that, as in the case of single disciplines, the adjoint approach is preferred when the number of design variables is significantly larger than the number of functions of interest. An alternate adjoint formulation, which had not been previously published, was introduced.

The sensitivities computed by the lagged-coupled adjoint method were compared to sensitivities given by the complex-step derivative approximation and shown to be extremely accurate, having an average relative error of 2% for the cases tested. Comparison with single-discipline sensitivities showed that the true fully-coupled sensitivities are essential for performing aero-structural design optimization.

In realistic aero-structural design problems with hundreds of design variables, there is a considerable reduction in computational cost when using the coupled-adjoint method as opposed to either other analytic approaches — such as the coupled-direct method — or approaches with easier implementations, such as finite differencing and the complex-step approximation. This advantage is due to the fact that the cost associated with the adjoint method is almost independent of the number of design variables.

The coupled sensitivities computed using the methodology developed in this dissertation were successfully employed to solve a proof-of-concept aircraft design problem for the case of a supersonic business jet configuration. This configuration was parameterized with a large number of aerodynamic and structural variables, and two different flight conditions were considered: a cruise condition for which the drag coefficient was minimized and a maneuver condition for which the structural stress constraints were enforced. The outcome of this optimization was compared with the traditional method of sequential optimization and it was found that the integrated approach yielded a better design.

Future work in the development of the aero-structural design framework is expected to add further capabilities such that more realistic aircraft design problems can be solved. To achieve this, more than two flight conditions should be considered, and the current infrastructure allows it. Some of the additional flight conditions would be associated with aerodynamic performance. In the case of the natural laminar flow supersonic business jet, for example, it would be extremely useful to include a subsonic cruise condition for overland flight. Other flight conditions would be critical aerodynamic load cases associated with the structural stress constraints. Additional non-aerodynamic load cases such as taxi bumps could also be considered.

Another interesting improvement to the aero-structural design framework would involve adding of more shape and wing planform design variables to the parameterization of aircraft configurations as well as increasing in the complexity of the geometry to include diverters, nacelles, and empennage.

Although the KS function that was used to lump the stress constraints seemed to work well in the design case that was presented, more research is needed to determine the consequences of using this lumping function, namely how it affects the efficiency of the optimization process and the accuracy of the optimum result.

The development of the complex-step derivative approximation proved to be crucial in achieving the main goal of this dissertation. In one particular case, an error in the implementation of the lagged-coupled adjoint was found only because the benchmark results given by the complex-step estimates were accurate enough to make this error obvious, when comparisons with finite-difference results were inconclusive.

The research into the application of the complex-step method to real-world numerical algorithms yielded several new insights. Solutions to subtle problems in its implementation were clarified, and its relationship to traditional algorithmic differentiation methods was exposed. This enabled the application of a substantial body of knowledge — that of the algorithmic differentiation community — to the complex-step method, answering many important questions.

The implementation process of the complex-step derivative approximation was successfully automated for two large solvers, including an iterative one. The resulting derivative estimates were validated by comparison with results obtained with better-known methods.

The complex-step method, unlike the finite-difference approximations, was shown to have the advantage of being step size insensitive and for small enough steps, the accuracy of the sensitivity estimates was only limited by the numerical precision of the algorithm.

The examples presented herein illustrate these points and put forward two excellent uses for the method: validating a more sophisticated gradient calculation scheme and providing accurate and smooth gradients for analyses that accumulate substantial computational noise.

Appendix A

Low-Fidelity Aero-Structural Optimization

This appendix describes how the results shown in Figure 1-6 were obtained. This study was carried out in order gain a better understanding of the different coupled sensitivity analysis methods and to learn about the implementation issues on a simple problem before dealing with the added complexity of the high-fidelity aero-structural solver.

A.1 Governing Equations

The structural model is composed of a single wing spar with circular cross-section which is modeled by tubular beam finite elements. The structural finite-element equations can be written as

$$Ku - f = 0 \tag{A.1}$$

where K is stiffness matrix of the structure, u is the vector of displacements and rotations (6 degrees of freedom per node) and f is the vector of external forces. The spar geometry and discretization is shown in Figure A-1.

The aerodynamic analysis is performed by a panel code, which models a wing and solves the linear system

$$A\Gamma - v = 0, \tag{A.2}$$

where A is the aerodynamic influence coefficients matrix, Γ is the vector of panel circulations and v is the vector of panel boundary conditions, which is simply the local angle of attack of each panel. Figure A-1 shows how the panels are distributed on the wing.

When solving the aero-structural system, we also want the wing to produce the lift required to maintain level flight, i.e.,

$$L - W = 0, \tag{A.3}$$

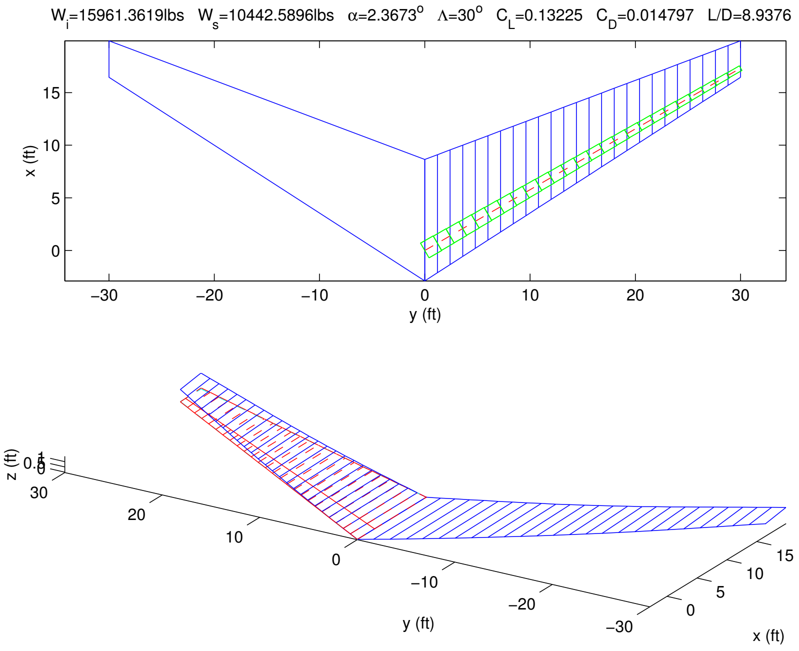

Figure A-1: Aerodynamic and structural discretization of the wing, showing structural displacements. Note that the structure consists of a tubular spar.

where L is the total wing lift and W is the total weight of aircraft. This requirement can be satisfied by setting the angle of attack, α, to the appropriate.

The simultaneous solution of these three sets of equations (A.1–A.3) defines the state of the aero-structural system. The state variables of the coupled system are u, Γ, and α. The matrices A, K and the aircraft weight, W are constant. Note that the remaining variables that are not state variables have the following dependencies,

$$v = v(u, \alpha), \tag{A.4}$$

$$f = f(\Gamma), \tag{A.5}$$

$$L = L(\Gamma). \tag{A.6}$$

A schematic representation of these dependencies is shown in Figure A-2.

We now want to use Newton's method to solve the simultaneous equations (A.1–A.3). Using calculus of variations we can take the total variation of each governing equation. When the system is solved, each governing equation equals zero. Then, any variation of the governing equations must also be zero for the system to remain solved. Starting with the structural equations,

$$(Ku - f) + \delta\,(Ku - f) = 0 \Rightarrow \tag{A.7}$$

$$K\delta u - \frac{\partial f}{\partial \Gamma}\delta\Gamma = f - Ku. \tag{A.8}$$

Here we used the fact that the forces do not depend directly on the displacements or the angle of attack. For the aerodynamic equations we obtain,

$$(A\Gamma - v) + \delta\,(A\Gamma - v) = 0 \Rightarrow \tag{A.9}$$

$$A\delta\Gamma - \frac{\partial v}{\partial u}\delta u - \frac{\partial v}{\partial \alpha}\delta\alpha = v - A\Gamma. \tag{A.10}$$

Finally, for the lift constraint equation, we find that

$$(L - W) + \delta\,(L - W) = 0 \Rightarrow \tag{A.11}$$

$$\frac{\partial L}{\partial \Gamma}\delta\Gamma = W - L. \tag{A.12}$$

Writing these equations in matrix form we obtain,

$$\begin{bmatrix} K & -\frac{\partial f}{\partial \Gamma} & 0 \\ -\frac{\partial v}{\partial u} & A & -\frac{\partial v}{\partial \alpha} \\ 0 & \frac{\partial L}{\partial \Gamma} & 0 \end{bmatrix} \begin{bmatrix} \delta u \\ \delta\Gamma \\ \delta\alpha \end{bmatrix} = \begin{bmatrix} f - Ku \\ v - A\Gamma \\ W - L \end{bmatrix}. \tag{A.13}$$

This constitutes the Newton method for coupled systems. Since the system is linear, the Newton iteration converges a single step.

In addition to solving for the state of the system, the aero-structural analysis module also computes the total drag of the wing (D), the structural weight (W_s) and the range (R) using the Breguet range equation

$$R(L, W_s) = \frac{V}{c}\frac{L}{D}\ln\frac{W_i}{W_s + W_0}, \tag{A.14}$$

where V is the cruise velocity and c is the thrust specific fuel consumption of the powerplant. L/D is the ratio of lift to drag, and $W_i/(W_s + W_0)$ is the ratio of initial and final cruise weights of the aircraft. Note that the final cruise weight is the structural weight plus a constant weight.

A.2 Coupled Sensitivity Analysis

The sensitivities of drag coefficient and structural stresses with respect the wing twist distribution and finite-element thicknesses were computed using both the direct and adjoint methods described in Chapter 4. Based on the notation of that chapter, we set

$$\mathcal{R}_{k'} = \begin{bmatrix} Ku - f \\ \\ A\Gamma - v \\ \\ L - W \end{bmatrix}, \quad y_{i'} = \begin{bmatrix} u \\ \\ \Gamma \\ \\ \alpha \end{bmatrix}. \tag{A.15}$$

The matrix of sensitivities of the governing equation residuals with respect to the state variables of the system is the same as the matrix used in the Newton iteration (A.13), i.e.,

$$\frac{\partial \mathcal{R}_k}{\partial y_i} = \begin{bmatrix} K & -\frac{\partial f}{\partial \Gamma} & 0 \\ -\frac{\partial v}{\partial u} & A & -\frac{\partial v}{\partial \alpha} \\ 0 & \frac{\partial L}{\partial \Gamma} & 0 \end{bmatrix} \tag{A.16}$$

We can now use either the direct or adjoint methods to calculate the total sensitivity of the drag or structural stress using equation (4.6).

A.3 Optimization Results

To optimize this design, we chose to maximize the range. A simplified aircraft design optimization problem can be stated as follows:

$$\begin{aligned} \text{maximize} \quad & R(D(\gamma, t), W_s(t)) \\ \text{with respect to} \quad & \gamma, t \\ \text{subject to} \quad & L = W \\ & \sigma \leq \sigma_{\text{yield}} \\ & t \geq t_{\min}. \end{aligned} \tag{A.17}$$

Performance		
Cruise Mach number	0.8	
Range	3,000	nm
Cruise altitude	40,000	ft
Cruise TSFC	0.58	

Wing geometry		
Span	60	ft
Sweep	30	degrees
Aspect ratio	8	
Taper ratio	0.3	

Table A.1: Specifications for the small transonic business jet.

The design variables include the jig twist distribution of the wing (γ), and the wall thicknesses of the tube finite-elements that form the wing spar (t). The drag corresponds to the initial cruise condition and the stresses correspond to a 2.5g maneuver condition.

To demonstrate this simple aircraft design methodology, we chose to optimize the wing of a typical business jet. The specifications for this particular configuration are listed in Table A.1. The results for the baseline configuration are shown in Figure A-3. Note that linear distributions are set for both the jig twist of the wing an the wall thicknesses of the tubular finite elements. From the twist distribution of the deflected wing, we can see that the wing twists down, which is expected for a swept configuration. The values for all the finite-element stresses are below the yield stress (represented by the red line).

The results obtained by solving the optimization problem (A.17) are shown in Figure A-4. The optimized structure is thickest at the root, and the reaches the minimum gauge at around 20 ft from the root. The finite elements are fully stressed up to this same point. The optimization problem (A.17) was also solve without structural variables for a rigid wing. The results of this case are depicted in Figure A-5. The optimized lift distribution matches the elliptic distribution exactly.

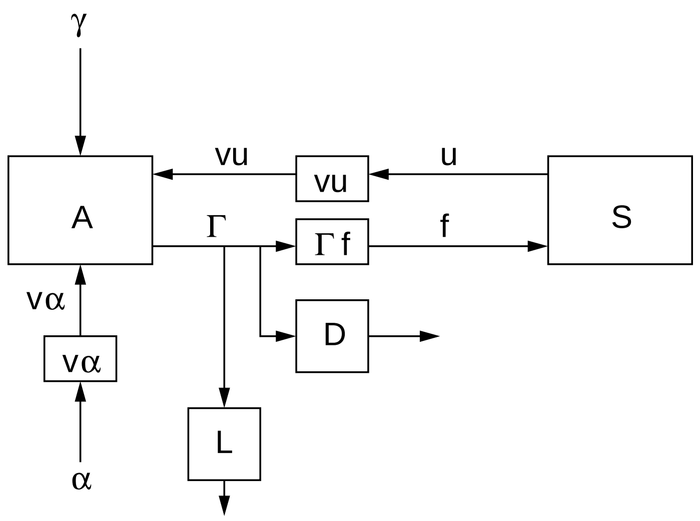

Figure A-2: Schematic representation of the aerodynamic and structural equations, showing the coupling between the two and the variable dependencies.

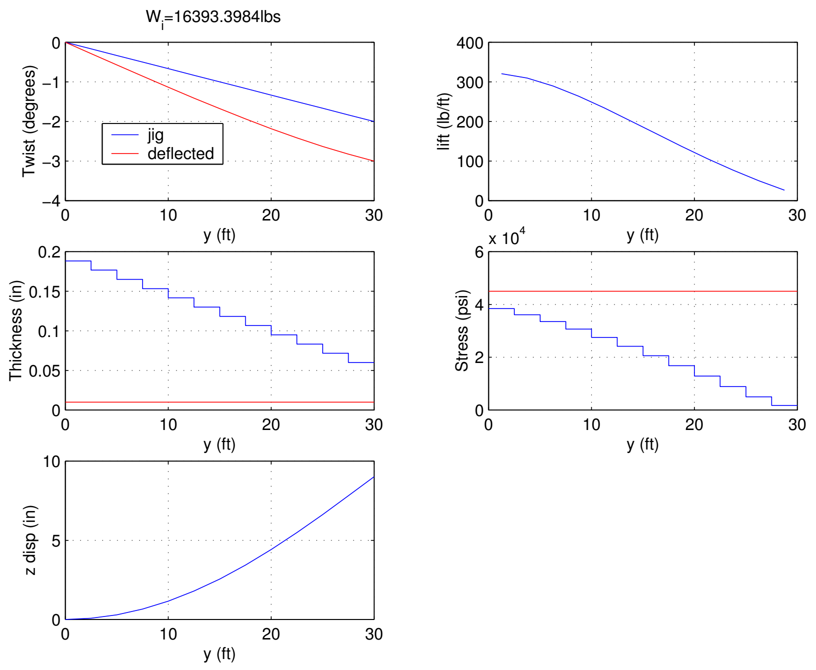

Figure A-3: Baseline twist, tube thickness, vertical displacement, lift, and stress distributions.

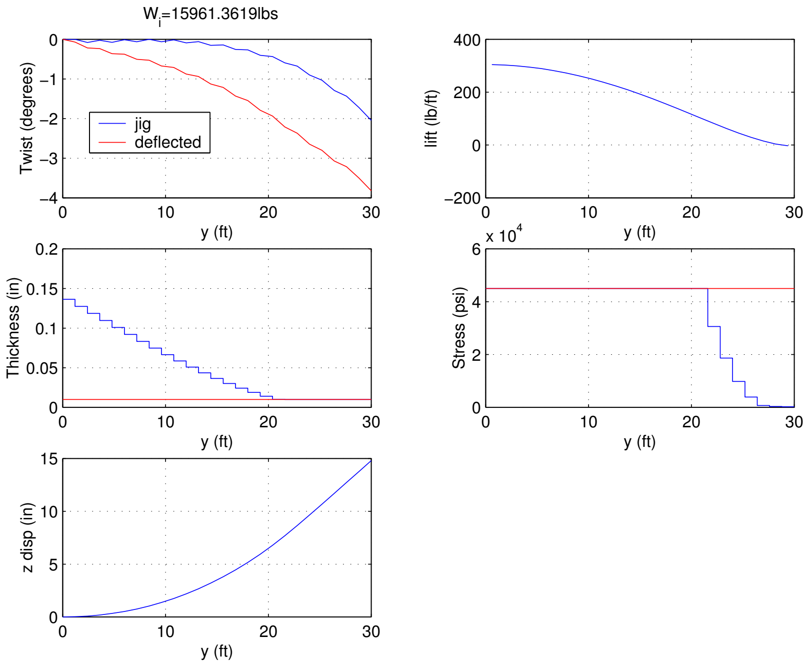

Figure A-4: Optimized twist, tube thickness, vertical displacement, lift, and stress distributions.

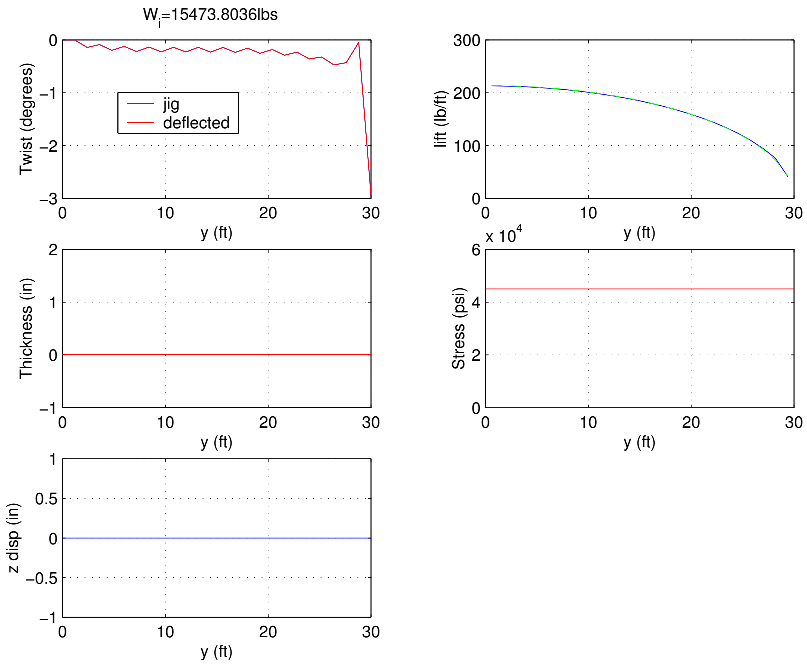

Figure A-5: Optimized twist, tube thickness, vertical displacement, lift, and stress distributions for a rigid wing. Note that the optimized lift distribution is exactly elliptic.

Bibliography

[1] H. M. Adelman and R. T. Haftka. Sensitivity analysis of discrete structural systems. *AIAA Journal*, 24(5):823–832, May 1986.

[2] M. A. Akgün, R. T. Haftka, K. C. Wu, and J. L. Walsh. Sensitivity of lumped constraints using the adjoint method. *AIAA Paper* 99-1314, Apr. 1999.

[3] N. Alexandrov and M. Y. Hussaini, editors. *Multidisciplinary Design Optimization: State-of-the-Art*. SIAM, 1997.

[4] N. M. Alexandrov. Multilevel methods for MDO. In *Multidisciplinary Design Optimization: State-of-the-Art*. SIAM, 1997.

[5] W. K. Anderson, J. C. Newman, D. L. Whitfield, and E. J. Nielsen. Sensitivity analysis for the Navier–Stokes equations on unstructured meshes using complex variables. *AIAA Paper* 99-3294, 1999.

[6] T. Beck. Automatic differentiation of iterative processes. *Journal of Computational and Applied Mathematics*, 50:109–118, 1994.

[7] T. Beck and H. Fischer. The if-problem in automatic differentiation. *Journal of Computational and Applied Mathematics*, 50:119–131, 1994.

[8] K. G. Bhatia and J. Wertheimer. Aeroelastic challenges for a high speed civil transport. *AIAA Paper* 93-1478, Feb. 1993.

[9] C. Bischof, A. Carle, G. Corliss, A. Grienwank, and P. Hoveland. ADIFOR: Generating derivative codes from Fortran programs. *Scientific Programming*, 1(1):11–29, 1992.

[10] C. Bischof, G. Corliss, and A. Grienwank. ADIFOR exception handling. Technical Report MCS-TM-159, Argonne Technical Memorandum, 1991.

[11] C. H. Bischof, L. Roh, and A. J. Mauer-Oats. ADIC: an extensible automatic differentiation tool for ANSI-C. *Software — Practice and Experience*, 27(12):1427–1456, 1997.

[12] R. Braun and I. Kroo. Development and application of the collaborative optimization architecture in a multidisciplinary design environment. In *Multidisciplinary Design Optimization: State of the Art*. SIAM, 1996.

[13] S. A. Brown. Displacement extrapolation for CFD+CSM aeroelastic analysis. *AIAA Paper* 97-1090, Jan. 1997.

[14] C. Faure and Y. Papegay. *Odyssée Version 1.6. The language reference manual*. INRIA, 1997. Rapport Technique 211.

[15] G. Corliss, C. Faure, A. Griewank, L. Hascoet, and U. Naumann, editors. *Automatic Differentiation: From Simulation to Optimization*. Springer, 2001.

[16] E. J. Cramer, J. E. Dennis, P. D. Frank, R. M. Lewis, and G. R. Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal on Optimization*, 4:754–776, Nov. 1994.

[17] A. DeMiguel and W. Murray. An analysis of collaborative optimization methods. *AIAA Paper* 2000-4720, Sept. 2000.

[18] S. A. Forth and T. P. Evans. *Aerofoil Optimisation via Automatic Differentiation of a Multigrid Cell-Vertex Euler Flow Solver*, chapter 17, pages 149–156. Springer, 2001.

[19] R. Giering. Tangent linear and adjoint model compiler users manual, 1997.

[20] P. E. Gill, W. Murray, and M. A. Saunders. *User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming*. Systems Optimization Laboratory, Stanford University, California, 94305-4023, Nov. 1994. Technical Report SOL 94-1.

[21] P. E. Gill, W. Murray, M. A. Saunders, and M. A. Wright. *User's Guide for SNOPT 5.3: A Fortran Package for Large-scale Nonlinear Programming*. Systems Optimization Laboratory, Stanford University, California, 94305-4023, Dec. 1998. Technical Report SOL 98-1.

[22] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, San Diego, CA 92101, 1981.

[23] A. A. Giunta. A novel sensitivity analysis method for high fidelity multidisciplinary optimization of aero-structural systems. *AIAA Paper* 2000-0683, Jan. 2000.

[24] A. Griewank. Personal communication, Aug. 1998.

[25] A. Griewank. *Evaluating Derivatives*. SIAM, Philadelphia, 2000.

[26] A. Griewank, C. Bischof, G. Corliss, A. Carle, and K. Williamson. Derivative convergence for iterative equation solvers. *Optimization Methods and Software*, 2:321–355, 1993.

[27] A. Griewank, D. Juedes, and J. Utke. Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software*, 22(2):131–167, June 1996.

[28] M. E. Holden. *Aeroelastic Optimization Using the Collocation Method*. PhD thesis, Stanford University, Stanford, CA 94305, 1999.

[29] J. E. Horwedel. *GRESS Version 2.0 User's Manual*. ORNL, Nov. 1991. TM-11951.

[30] A. Jameson. Aerodynamic design via control theory. *Journal of Scientific Computing*, 3:233–260, 1988.

[31] A. Jameson. Automtatic design of transonic airfoils to reduce the shock induced pressure drag. In *Proceedings of the 31st Israel Annual Conference on Aviation and Aeronautics*, Tel-Aviv, Feb. 1990.

[32] A. Jameson. Re-engineering the design process through computation. *AIAA Paper* 97-0641, Jan. 1997.

[33] A. Jameson, L. Martinelli, and N. A. Pierce. Optimum aerodynamic design using the Navier–Stokes equations. *Theoretical and Computational Fluid Dynamics*, 10:213–237, 1998.

[34] A. Jameson, W. Schmidt, and E. Turkel. Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes. *AIAA Paper* 81-1259, June 1981.

[35] S. Kodiyalam and J. Sobieszczanski-Sobieski. Bilevel integrated system synthesis with response surfaces. *AIAA Journal*, 38(8):1479–1485, aug 2002.

[36] I. Kroo, R. Tracy, J. Chase, and P. Sturdza. Natural laminar flow for quiet and efficient supersonic aircraft. *AIAA Paper* 2002-0146, Jan. 2002.

[37] I. M. Kroo. Decomposition and collaborative optimization for large scale aerospace design. In *Multidisciplinary Design Optimization: State of the Art*. SIAM, 1996.

[38] M. Lutz. *Programming Python*. O'Reilly & Associates, Inc., Cambridge, MA 02140, Fall 1996.

[39] J. N. Lyness. Numerical algorithms based on the theory of complex variable. In *Proceedings — ACM National Meeting*, pages 125–133, Washington DC, 1967. Thompson Book Co.

[40] J. N. Lyness and C. B. Moler. Numerical differentiation of analytic functions. *SIAM Journal on Numerical Analysis*, 4(2):202–210, June 1967.

[41] J. N. Lyness and G. Sande. ACM Algorithm 413: ENTCAF and ENTCRE: Evaluation of normalized Taylor coefficients of an analytic function (C5). *Communications of the ACM*, 14(10):669–675, Oct. 1971.

[42] J. R. R. A. Martins, 2003. http://mdolab.utias.utoronto.ca.

[43] J. R. R. A. Martins, J. J. Alonso, and J. Reuther. Aero-structural wing design optimization using high-fidelity sensitivity analysis. In *Proceedings — CEAS Conference on Multidisciplinary Aircraft Design Optimization, Cologne, Germany*, pages 211–226, June 2001.

[44] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. Complete configuration aero-structural optimization using a coupled sensitivity analysis method. *AIAA Paper* 2002-5402, Sept. 2002.

[45] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. High-fidelity aero-structural design optimization of a supersonic business jet. *AIAA Paper* 2002-1483, Apr. 2002.

[46] J. R. R. A. Martins, I. M. Kroo, and J. J. Alonso. An automated method for sensitivity analysis using complex variables. *AIAA Paper* 2000-0689, Jan. 2000.

[47] J. R. R. A. Martins, P. Sturdza, and J. J. Alonso. The connection between the complex-step derivative approximation and algorithmic differentiation. *AIAA Paper* 2001-0921, Jan. 2001.

[48] K. Maute, M. Lesoinne, and C. Farhat. Optimization of aeroelastic systems using coupled analytical sensitivities. *AIAA Paper* 2000-0560, Jan. 2000.

[49] K. Maute, M. Nikbay, and C. Farhat. Coupled analytical sensitivity analysis and optimization of three-dimensional nonlinear aeroelastic systems. *AIAA Journal*, 39(11):2051–2061, Nov. 2001.

[50] K. I. M. McKinnon. Convergence of the Nelder-Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9:148–158, 1999. reccommended from AA222.

[51] J. C. Newman, W. K. Anderson, and L. Whitfield, D. Multidisciplinary sensitivity derivatives using complex variables. Technical Report MSSU-COE-ERC-98-08, Computational Fluid Dynamics Laboratory, July 1998.

[52] S. Obayashi and D. Sasaki. Self-organizing map of pareto solutions obtained from multiobjective supersonic wing design. *AIAA Paper* 2002-0991, 2002.

[53] F. W. J. Olver. *Error Analysis of Complex Arithmetic*, pages 279–292. 1983.

[54] J. D. Pryce and J. K. Reid. AD01, a Fortran 90 code for automatic differentiation. Report RAL-TR-1998-057, Rutherford Appleton Laboratory, Chilton, Didcot, Oxfordshire, OX11 OQX, U.K., 1998.

[55] J. Reuther, J. J. Alonso, A. Jameson, M. Rimlinger, and D. Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers: Part I. *Journal of Aircraft*, 36(1):51–60, 1999.

[56] J. Reuther, J. J. Alonso, A. Jameson, M. Rimlinger, and D. Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers: Part II. *Journal of Aircraft*, 36(1):61–74, 1999.

[57] J. Reuther, J. J. Alonso, J. R. R. A. Martins, and S. C. Smith. A coupled aero-structural optimization method for complete aircraft configurations. *AIAA Paper* 99-0187, 1999.

[58] J. Reuther, J. J. Alonso, J. C. Vassberg, A. Jameson, and L. Martinelli. An efficient multiblock method for aerodynamic analysis and design on distributed memory systems. *AIAA Paper* 97-1893, June 1997.

[59] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. *AIAA Paper* 96-0094, 34th Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1996.

[60] J. J. Reuther. *Aerodynamic Shape Optimization Using Control Theory.* PhD thesis, University of California Davis, May 1996. also NASA-CR-201064.

[61] E. B. Saff and A. D. Snider. *Fundamentals of Complex Analysis.* Prentice Hall, New Jersey, 1976.

[62] D. Sasaki, S. Obayashi, and K. Nakahashi. Navier–Stokes optimization of supersonic wings with four design objectives using evolutionary algorithm. *AIAA Paper* 2001-2531, 2001.

[63] J. Sobieszczanski-Sobieski. Sensitivity of complex, internally coupled systems. *AIAA Journal*, 28(1):153–160, Jan. 1990.

[64] J. Sobieszczanski-Sobieski. A technique for locating function roots and for satisfying equality constraints in optimization. *Structural Optimization*, 4(3–4):241–243, Oct. 1992.

[65] J. Sobieszczanski-Sobieski and R. T. Haftka. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural Optimization*, 14(1):1–23, 1997.

[66] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Review*, 40(1):110–112, Mar. 1998.

[67] R. Tracy, I. Kroo, J. Chase, and J. Viken. Exploiting natural laminar flow for economical supersonic transport. *SAE Paper* 952040, Sept. 1995.

[68] S. R. Wakayama. *Lifting Surface Design Using Multidisciplinary Optimization.* PhD thesis, Stanford University, Stanford, CA 94305, Dec. 1994.

[69] J. L. Walsh, K. C. Young, F. J. Tarzanin, J. E. Hirsh, and D. K. Young. Optimization issues with complex rotocraft comprehensive analysis. *AIAA Paper* 98-4889, Sept. 1998.

[70] J. Yao, J. J. Alonso, A. Jameson, and F. Liu. Development and validation of a massively parallel flow solver for turbomachinery flow. *Journal of Propulsion and Power*, 17(3):659–668, June 2001.

[71] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.