Complex System Optimization: A Review of Analytical Target Cascading, Collaborative Optimization, and Other Formulations

by

James T. Allison

A master's thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Mechanical Engineering
in The University of Michigan
2004

Master's Thesis Committee:

Professor Panos Papalambros,
Dr. Michael Kokkolaras

ABSTRACT

Complex System Optimization: A Review of Analytical Target Cascading, Collaborative Optimization, and Other Formulations

by
James T. Allison

Chair: Panos Papalambros

Design of some modern products requires special techniques to manage complexity. Various industries have specific needs in this regard, and several methodologies for *complex system optimization* have been developed in response. A critical review of these sometimes diverse approaches offers the design community enhanced resources for mapping approaches to present design problems. This thesis covers several selected single-level and multi-level methodologies for complex system optimization. Two novel and easily replicated engineering design example problems are introduced, facilitating an illustrative implementation of the said methodologies, and offering a venue for clear exposition of their distinctions.

Emphasis is given to two particular multi-level methodologies: *Analytical Target Cascading* (ATC), and *Collaborative Optimization* (CO). ATC was developed as a product development tool, and has ties to the automotive industry. CO is a Multidisciplinary Design Optimization formulation, evolved from established methods for Multidisciplinary Analysis. CO sees regular use in aerospace analysis and design

problems. The origin of each methodology colors its nature. Although ATC and CO emerged from different sources, their mathematical formulations appear to be similar. These formulations are investigated in detail, and it is shown that each has a unique solution process. Terminology for each formulation is clearly defined and compared.

This review is an important contribution toward better understanding of complex system optimization methodologies. This in turn helps advance industry acceptance and utilization of these methodologies.

To Natalie

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Panos Papalambros, for giving me the opportunity to join the ODE laboratory, and to learn from him and others in this group. Thanks to Dr. Michael Kokkolaras for the hours spent with me going through the details of complex systems optimization, and helping to guide the direction of this work. I would like to acknowledge the assistance of Brian Roth of Stanford University in reviewing this work and ensuring an accurate account of the Collaborative Optimization formulation.

I would especially like to thank my wife, Natalie, for offering enduring support of my studies. Finally, I thank my parents and other family members for encouraging me to live to my full potential.

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

CHAPTER 1

Introduction

Modern engineering products are becoming increasingly complex, particularly in industries such as aerospace and automotive. Optimization of complex products presents unique challenges, including the coordination of frequently competing design activities. Numerous formal methodologies have been developed for complex system optimization. This thesis provides background for understanding many of these methodologies, and offers a critical review of selected *single-level* and *multi-level* formulations.

Methodologies for complex system optimization are commonly developed based on the needs of a particular industry. The needs of different industries can be diverse, leading to methodologies with distinctly different objectives and processes. In spite of these differences, the resulting mathematical formulations can be similar, and in fact some methodologies can be coaxed into forms that appear to be nearly identical. However, the processes are still different, and the original intent of a methodology is inescapable, coloring its nature.

This chapter first discusses optimization of complex systems in a general context, and then how these ideas pertain to product development. An overview of the thesis follows.

1.1 Complex Systems Analysis

Design of a product classified as a complex system poses substantive challenges to both analysis and to design optimization. Specialized techniques have been developed to meet these challenges. This section provides a general overview of complex systems, and an introduction to their analysis.

1.1.1 Definition of Complex Systems

A complex system is:

> An assembly of interacting members that is difficult to understand as a whole.

The terms 'interaction' and 'difficult to understand' merit precise definition. An interaction between members exists if the state of one member affects how the system responds to changes in another member. A system is difficult to understand if an individual cannot understand the details of all members and all interactions between members. For example, aerospace products originally did not qualify as complex systems, since a single master designer was capable of completely understanding the design of an aircraft. Present aerospace products, however, are far too complex for an individual to understand. Several interacting specialized teams of experts are required to analyze and design a modern aircraft.

Assessment of whether a system is difficult to understand depends on available resources for investigation and analysis, which could include experience and intuition, mathematical models, empirical results, or computer simulations.

A system may qualify as complex due to its large scale (large number of members or inputs), or due to strong interactions. This thesis emphasizes systems with

complex interactions. These interactions are frequently more difficult to understand than the constituent members, but if well characterized provide opportunity to exploit synergy between members. When interaction is present, a system is said to be coupled; that is the members are coupled together via interaction.

Analysis of a complex system involves determining the system response for a given system design. The system design is completely determined by a set of design variables and parameters (system inputs). Design or optimization of a complex system involves determining the values for design variables that produce the best system response based on some criteria. Design variables can be changed by the designer, while design parameters are considered fixed during the design process [32]. System analysis is used as a tool for system design.

1.1.2 Complex System Decomposition

Frequently the analysis of complex systems as an undivided whole is discovered to be inefficient, if not intractable. An alternative strategy is to partition the system into smaller subsystems. Considering the subsystems individually and their interactions may render the system analysis task feasible or more efficient.

Several approaches for system partitioning may be used, and the choice depends on the system and the analysis environment. Wagner [44] identified four categories of system partitioning methods: by object, by aspect, sequential, or matrix. Object decomposition involves dividing a system by physical component or function [35]. For example, an automotive design may be partitioned by object into body, powertrain, and suspension subsystems. Aspect partitioning divides the system by discipline. The same automotive design could be partitioned by aspect into structural, aerodynamic, and dynamics disciplines. Sequential partitioning is appropriate for

flow processes. Chemical or manufacturing processes may be partitioned by making cuts in flow paths. Matrix partitioning is applied to large systems of mathematical equations. Wagner also explained that in a mathematical programming context, *decomposition is both partitioning and application of the coordination strategy.*

This thesis will focus on the first two types of partitioning: object and aspect. Much work has already been done in the area of system partitioning methodology [27, 44], i.e. formal approaches to partition systems in the best way possible. In this thesis an established system partition will be assumed to exist, and emphasis will instead be on formulation of strategies to fit an already partitioned system.

Several designations have been given to partitioned subsystems (disciplines, members, elements, components, subproblems, etc.). The designation used typically depends on the partitioning approach, the nature of the system, and the analysis environment. To circumvent possible obfuscation, these subsystems will be referred to as subspaces— a general term independent of problem and partitioning type. The original system has a system level space, consisting of the system's input and output spaces. Subsystems also have their own respective subspaces, consisting of their own input and output spaces. These subspaces in general include some of the original system level inputs and outputs, as well as interaction variables that are artifacts of system partitioning.

System partitioning is also characterized by the structure of its communication pathways. A non-hierarchical system has no restrictions on the communication pathways (Figure 1.1), while a hierarchical system only allows communication between parent and child subspaces as depicted in Figure 1.2. For example, subspace 2 and 3 are not allowed to exchange information in the hierarchical system example. Numerous systems possess a naturally hierarchic structure, which can be exploited to

facilitate efficient solution processes. Some formulations for complex system optimization discussed in later chapters take general non-hierarchic systems and repose them as hierarchic systems using additional system constraints.

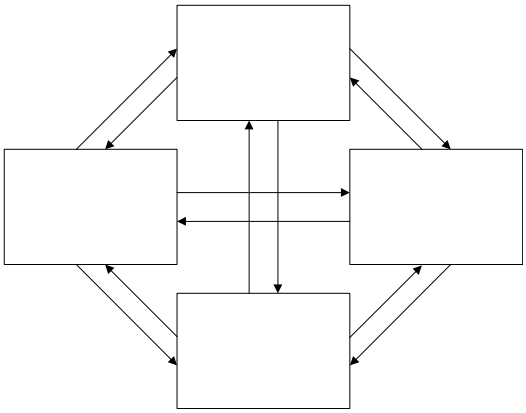

Figure 1.1: Illustration of a non-hierarchical system.

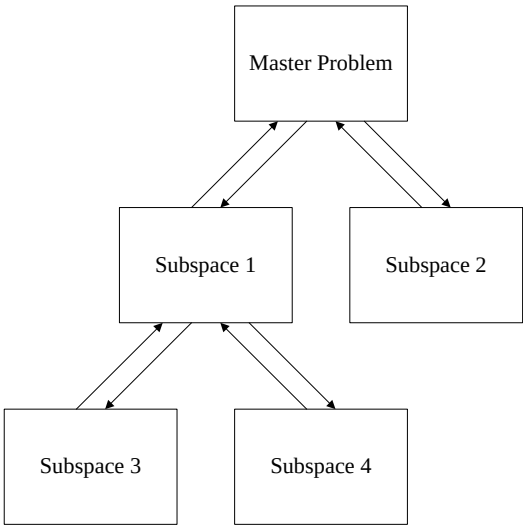

Figure 1.2: Illustration of a hierarchical system.

1.1.3 Subspace Interaction

As explained previously, an interaction between subspaces indicates that the state of one subspace affects how the system responds to a change in another subspace. This agrees with the classical Design of Experiments definition of interaction. If the

system response is the output of function f, and the inputs to subspaces 1 and 2 are X_1 and X_2 respectively[1], then the interaction between the two subspaces is:

$$\text{Int}(SS1, SS2) = [f(X_1^+, X_2^+) - f(X_1^-, X_2^+)] - [f(X_1^+, X_2^-) - f(X_1^-, X_2^-)]$$

Sometimes subspace interaction is straightforward, such as when communication flow is unidirectional (when one subspaces depends on another, but not visa-versa). Other more involved interactions occur when subspaces depend on each other. A few classical examples of this include aeroelasticity and combustion (both partitioned by aspect). In aeroelasticity the system is divided into the structural analysis and the aerodynamic analysis. In aircraft wing analysis, the structural analysis takes the pressures generated by aerodynamic effects as inputs and returns the wing deflections and stresses. The input to the structural analysis (pressure distribution) must be generated by an aerodynamic analysis. However, the aerodynamic analysis requires the output of the structural analysis (wing deflections) to accurately predict the pressure distribution. The structural and aerodynamic analyses are therefore said to be coupled— they each depend on the other. The other example, combustion, has even more complex interaction. Combustion requires the analysis of fluid transport, heat transfer, and chemical reactions. Each of the three disciplines depends upon the other two, resulting in a total of six couplings. If a system has n distinct subspaces, then the system has a possible $n(n-1)$ couplings[2]. The combustion example is fully coupled, since all of the possible $3(3-1)$ couplings exist. The strength of each coupling depends on many factors, discussed in §2.2. A strongly coupled system, even if it only has a few subspaces, may pose substantial analysis difficulty.

An illustrative example with two coupled subspaces is introduced here, and fully

[1]The superscripts $^+$ and $^-$ indicate two different levels of the inputs.

[2]Please observe that these couplings may be scalar, vector, or function valued.

developed in Section 4.1 for the demonstration of two fundamental complex system optimization strategies. A turbine blade is used in a gas-turbine engine to convert the kinetic energy of high-velocity combustion gasses to rotational mechanical work. Gas turbine engine design has been the subject of complex systems optimization research. Röhl et. al. demonstrate the use of several different complex system optimization strategies for the design of turbine engines and related manufacturing processes [34]. The simplified example problem here considers only the analysis and design of a single turbine blade. The objective is to minimize the heat loss through the blade, while meeting mass, stress and temperature constraints. The system analysis is partitioned by aspect into two subspaces: thermal and structural analysis. The thermal analysis requires the length of the blade to compute the heat loss and the temperature distribution, while the structural analysis requires the temperature distribution to compute the dilated blade length. The coupling of this system is illustrated in Figure 1.3.

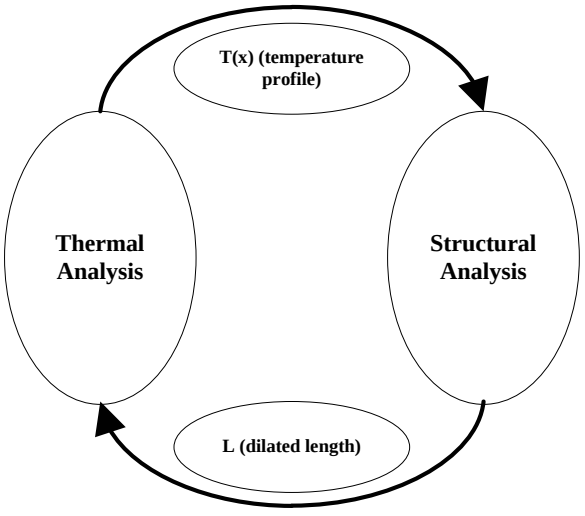

Figure 1.3: System coupling of the turbine blade example problem.

1.2 Product Development

Product development is an important application of complex system optimization. Needs within industry drive the development of approaches to improve the design process, and a complex systems approach is sometimes utilized. A better understanding of these approaches among all involved in the development of complex products is critical to the improvement of the design process.

1.2.1 Design of Complex Systems

Design is a decision making process: decisions must be made about the appropriate values of design variables to yield the most optimal design. These decisions may be based informally on intuition, or formally on quantitative performance criteria. An introduction to formal design optimization is given in Section 2.1. This section discusses design optimization as it relates to product development.

Product design optimization requires an analysis method and a method for generating improved designs. The spectrum of available optimization procedures ranges from iterative trial-and-error to formal algorithms to efficiently guide the design process to an optimal solution. An example of the former is prototype generation in a machine shop without formal preliminary design (design method), and experimental testing of the prototypes (analysis method). An example of the latter is a complete computer simulation of an automobile (analysis method) coupled with optimization software (design method). In either case, a better design is iteratively proposed based on the performance of previous designs. Trial-and-error approaches are adequate for simple, inexpensive designs. If prototypes are expensive or time-intensive to construct, it is desirable to use a simulation as the analysis tool. Whether experimentation or simulation is used for analysis, optimization algorithms can speed the

design process.

Product development has become increasingly complex. Market competitiveness requires ever increasing levels of product quality, reliability, and performance; traditional approaches to product development can be inadequate to meet these demands. Complex products, such as found in the aerospace or automotive industries, require teams of designers collaborating effectively to successfully generate competitive products. Creative individuals in industry have crafted novel approaches to product development, and researchers have formalized these approaches and work to improve their efficiency and robustness. Sharing knowledge of these approaches across industry boundaries will facilitate even more effective solutions to complex system optimization.

Several industries have adopted a multi-disciplinary framework to cope with the demands of designing complex systems, and value designers that thrive in a multi-disciplinary systems environment [8]. This gives impetus to a more ubiquitous familiarity with concepts discussed in this thesis. Some universities have included the study of complex system optimization in their curriculum [29], and ABET has placed increased emphasis on a multi-disciplinary systems perspective. Education of students, managers, and executives concerning the potential of complex system optimization methodologies will foster better acceptance and utilization in industry.

Industries are structured in a variety of ways, and as a result different industries spawn different approaches. One reason for this is because partitioning structure is a profound factor in how a problem can be solved. Partitioning might be enforced by established communication paths or by available analysis tools. Partitioning may also be dictated by heterogeneous computing environments and geographically separated design groups.

Matrix organizations may be aligned either by object or aspect[3] [43]. This existing structure leads to either object or aspect partitioning (respectively) of the design optimization problem. The task is then to formulate a strategy that fits the existing structure well and is efficient and effective compared to how the organization previously coordinated its product development tasks.

1.2.2 Industry Requirements

An important measure of complex system optimization strategy is its usefulness in industrial application. This section introduces some industrial needs that can be used to gauge the usefulness of strategies. Industry needs drive the formulation of these strategies. An effective complex system optimization strategy should:

- Compress design cycle time
- Facilitate transition to a concurrent engineering environment
- Characterize and exploit complex subspace interactions
- Yield substantive product quality and performance gains
- Exhibit flexibility and adaptability
- Advance product knowledge to beginning of the design process, while extending design freedom toward the end
- Effectively utilize existing resources

Reducing time to market is a critical metric for the success of a product. This can be realized by parallel utilization of resources, such as design groups and computing facilities. Sequential approaches can result in unproductive downtime. Parallelization also facilitates early consideration of manufacturing, reliability, and life cycle costs— a step towards a concurrent engineering environment and improved product

[3]For example, aerospace firms are typically aligned by aspect, and automotive firms are usually aligned by object.

quality. Understanding of subspace interactions enables exploitation of synergy between subspaces and sometimes dramatically improves system performance. Some products are complex enough that interactions and the system as a whole are impossible to understand using traditional knowledge based approaches [29], requiring the development of more formal strategies. As knowledge is gained about a system progressively throughout the design process, detail and complexity of the design increases. Analysis tools may also change as required by this increase in complexity. Design strategies must be flexible enough to adapt to changes and increase in complexity as the design process progresses [19].

Better design decisions are made when designers have improved knowledge of the system at hand, and freedom to act on this knowledge. Figure 1.4, adapted from [17], chronologically illustrates this objective. A preferred design approach is show with the dashed lines— indicating improved system knowledge and design freedom.

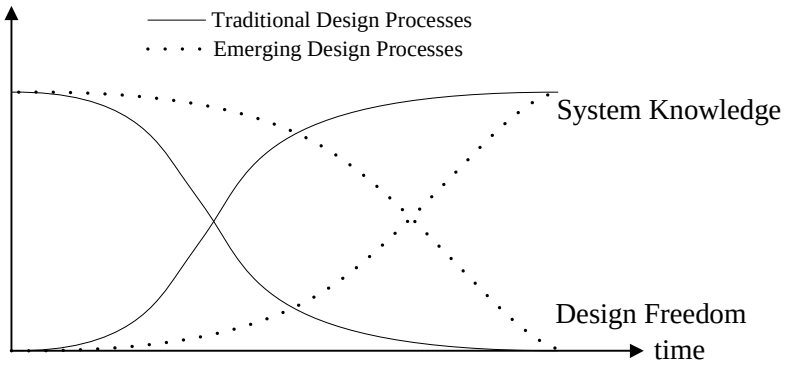

Figure 1.4: Illustration of maintaining design freedom and advancing system knowledge using emerging optimization strategies.

A short example illustrates how traditional sequential design processes can delay system knowledge acquisition and stifle design freedom. Consider a hypothetical

automotive design process, divided into the following sequential design tasks:

$$StructuralDesign \rightarrow PowertrainDesign \rightarrow SuspensionDesign \rightarrow ErgonomicsDesign$$

As the design progresses, one design group fixes its design (optimal according to its own criteria). Subsequent design groups must cope with these decisions by adjusting their own designs to compensate. Fixing design variables before all design tasks are considered may be detrimental to both subspace designs and to system level design objectives. If one of the design groups discovers their design is rendered infeasible by preceding design decisions, the process must be repeated. Interaction between the groups is dealt with, but not understood and exploited, passing up potential performance gains. For example, by the time the ergonomics design group begins work, the envelope of the cockpit has been fixed by the structural design group. It may be discovered that changing the envelope by just a few centimeters could dramatically improve the performance of the cockpit design, and that this small change has little effect on structural performance. Unfortunately this interaction cannot be exploited in a sequential design, since the structural design is fixed before the cockpit design begins. Parallelization enables consideration of interactions (advancing system knowledge) when the freedom still exists to act on this knowledge. Another reason system knowledge must be generated earlier is projects in some industries (such as aerospace) are farther and fewer between. Designers cannot rely on recent experience to guide the design process.

Mature industries have well developed design resources, including specialized analysis and optimization tools, and experienced designers and disciplinary experts. A desirable attribute of complex system optimization tools is the ability to make full use of these existing resources. Ideally designers are not removed from the process,

but are instead provided with insightful system information. This is a move toward a synergistic relationship between designers and computational tools.

Formal strategies for complex system design can be implemented at a range of points chronologically within the product development process. For example, Analytical Target Cascading (Chapters 5 and 7) is intended for use early in the process. Analytical Target Cascading takes system level targets and propagates them through the entire system, setting targets for all design groups such that each group can work independently toward a consistent system design that matches the original system level targets. Other methods may be employed early in the design process to optimize conceptual designs to their full potential before judgment between the alternatives is made. Some formulations are complete design tools, employing formal methods throughout the entire product development process, and adapting to increasing levels design detail.

1.2.3 Multidisciplinary Design Optimization

The aerospace industry deals with design of very complex systems, and aerospace firms are typically aligned by discipline (aspect). Researchers in this industry (and other institutions) have developed a class formal approaches to complex system design, referred to as Multidisciplinary Design Optimization (MDO). MDO approaches have roots in the field of Multidisciplinary Analysis (techniques for analyzing complex systems partitioned by discipline). MDO may be defined as both a new engineering discipline concerned with formal methods for the design of complex systems, and an environment conducive to said formal methods. The AIAA MDO Technical Committee [17] defines MDO as:

> A methodology for the design of complex engineering systems and sub-

systems that coherently exploits the synergism of mutually interacting phenomena.

Some are quick to identify that industry has always in some manner utilized the above approach. This is certainly true; MDO just formalizes the iteration and coordination between groups, improving efficiency and product quality.

An MDO environment facilitates effective communication between the appropriate groups. This alone is an important research focus. How should data formats be standardized? How should data be stored and distributed? The goal of an MDO environment is to have a tight-knit interdisciplinary team [17]. The communication process should not cause delays in product development. Industry has always at some level implemented MDO, but the development of more formal strategies enables industry to produce products with substantially better quality and reduced development time. It is not claimed that MDO introduces multidisciplinary coordination as a new endeavor, but rather offers a formalization of iteration and coordination between groups, improving efficiency and resulting in superior products.

The literature is rich with detailed explanations of MDO formulations and applications. Sobieski and Haftka provide a thorough review of developments in the MDO area [36]. The literature often tends toward specialization, and currently a paucity of solid overviews exists. A decade previous Balling and Sobieski published an excellent review of MDO formulations [5], and Balling and Wilkinson demonstrated several formulations with an analytical test problem [3]. Substantial developments have been made since these overviews, such as Collaborative Optimization (Chapters 5 and 6). The differences between some formulations are sometimes difficult to identify, leaving many to wonder about the merits of further pursuit. An opportunity exists to make the distinction between formulations, and clarify why they are all

important and what niches of the complex system optimization spectrum they fill.

1.3 Thesis Overview

This chapter introduced the basic concepts of complex system optimization and how it applies in a product development context. This thesis has two primary objectives: first, provide a solid background for understanding the basics of MDO and other strategies for complex system optimization; second, offer a critical review of selected *single-level* and *multi-level* formulations. Emphasis will be placed on clearly illustrating the differences between two popular strategies: Collaborative Optimization (CO) and Analytic Target Cascading (ATC).

Chapter 2, MDO Preliminaries, introduces a formal approach to systems optimization, and explains fixed point iteration (FPI), a popular approach to analyzing coupled systems. FPI is simple and intuitive to implement, yet has several shortcomings, including convergence difficulties. Knowledge of FPI enables a detailed understanding of the most fundamental MDO approach, MDF, and motivates the development of more sophisticated methods, such as IDF, AAO, and DCF— all of which are explained in Chapter 3.

Throughout this thesis illustrative examples are employed to establish important concepts. These examples are fully analytic, and presented in enough detail for straightforward replication. The turbine blade design problem introduced in this chapter is fully developed in Chapter 4, and used to illustrate the details of MDF and IDF implementation. In addition, it facilitates an interesting comparison between these two approaches.

The two multilevel methodologies of interest in this thesis, CO and ATC, are explained in Chapter 5. A structural analysis example problem was developed that

can be manipulated into a non-hierarchic form conducive to the CO formulation, and into a hierarchic form well suited for ATC. With such an example problem both CO and ATC can be used in a natural way to solve the same design problem, offering a unique comparison tool. The example problem is presented in Chapter 6 and solved using CO, and then reformulated in Chapter 7 and solved using ATC. Chapter 8 summarizes the comparison results, and offers insights into future related work.

Two themes flow through this thesis: exploiting interactions and proper application of approaches.

1. Complex systems have interdependent subspaces. A traditional sequential design process merely copes with the interactions, while modern strategies for complex system design seek to understand these interactions and exploit their synergy to generate superior designs.

2. Formulation of an effective approach for complex system optimization is highly problem dependent. The existing analysis structure and nature of subspace interactions are critical factors in how well suited particular approaches are for the design problem. A practitioner has a wide variety of formulations in his/her toolbox, and each fills a specific niche in the spectrum of complex system design problems. A thorough understanding of these formulations (not just the formulations from his/her industry) is crucial to success.

CHAPTER 2

MDO Preliminaries

Before embarking on an explanation of MDO formulations, several important system analysis and optimization concepts are presented. Formalized design optimization is reviewed, and critical aspects of coupled system analysis are explained. A solid understanding of these topics is required before strategies for complex system optimization are discussed, and is particularly important to appreciate the nuances and distinctions between strategies.

2.1 Systems Optimization

An introduction to the systems optimization paradigm is presented here, and nomenclature used throughout this thesis is developed.

2.1.1 Nonlinear Programming

The objective of nonlinear programming (NLP) is to find a vector $\mathbf{X}$ that minimizes (or maximizes) a function of that vector, and in the general case is subject to equality and inequality constraints. This applies to product design in that a design problem can be formalized into a nonlinear programming problem. It is assumed here that the conceptual design is already established, analysis tools exist, and the design can be completely described by a set of values contained in the design vector

$\mathbf{X}$. With the appropriate analysis tools, the merit of a particular design may be quantitatively established.

To illustrate, it might be desired to minimize the cost of a design, subject to performance constraints. The *objective* of the design problem is to minimize cost. In general the objective function is denoted $f(\mathbf{X})$. Any constraints may be expressed in negative null form. If a constraint specifies that some performance value $p(\mathbf{X})$ must not drop below a minimum acceptable value p_{min}, we can express this inequality constraint as $g(\mathbf{X}) = p_{min} - p(\mathbf{X}) \leq 0$. If a target value T must be achieved by a response function $T(\mathbf{X})$, this equality constraint may be written in the form $T - T(\mathbf{X}) = 0$. These are the negative null forms of the constraints. In general inequality constraint functions are denoted by $g(\mathbf{X})$, and equality constraint functions by $h(\mathbf{X})$. If multiple constraints are required, vector valued constraint functions are employed. A design optimization problem can in general be posed in a canonical form as a negative null nonlinear programming problem [6, 32]. The standard design problem is shown in equation 2.1.

$$\begin{aligned} \min_{\mathbf{X}} \quad & f(\mathbf{X}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{X}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{X}) = \mathbf{0} \end{aligned} \tag{2.1}$$

The general idea is to vary the values in $\mathbf{X}$ (i.e., search the set of available designs) such that the minimum value of $f(\mathbf{X})$ is obtained without violating any of the constraints $\mathbf{g}(\mathbf{X})$ or $\mathbf{h}(\mathbf{X})$. The solution to the problem is the *optimal design*, and is denoted $\mathbf{X}_*$. Constraints are said to be active if their removal changes the solution. This is a formal, quantitative way of posing the design optimization problem. An intuitive approach applicable to one, two, or perhaps three variable problems is to plot the objective and constraint functions over the design space, and visually identify

the optimum design point. This may work fine for a limited number of variables and easy to evaluate functions, but larger design problems and those with functions that take significant time to evaluate (prohibiting the generation of plots) require more efficient approaches.

Several algorithms exist to solve the general nonlinear programming problem in equation 2.1. When $\mathbf{X}$ is continuous, mathematical optimality conditions exist and can be used both to verify optimality and to drive gradient based algorithms. Examples of gradient based algorithms are Sequential Quadratic Programming (SQP) [6, 11, 32, 33] and Generalized Reduced Gradient (GRG) [6, 11, 32] methods. The well known Karush-Kuhn-Tucker (KKT) conditions (equation 2.2) are first order necessary conditions for optimality[1]. Proof of these conditions and delineation of optimality sufficiency conditions can be found in [6, 32, 41].

$$\begin{gathered} \nabla L = \nabla f(\mathbf{X}) + \nabla \lambda^T \mathbf{g}(\mathbf{X}) + \nabla \mu^T \mathbf{h}(\mathbf{X}) = \mathbf{0} \\ \mathbf{g}(\mathbf{X}) \leq \mathbf{0} \\ \mathbf{h}(\mathbf{X}) = \mathbf{0} \\ \lambda \neq \mathbf{0} \\ \mu \geq \mathbf{0} \end{gathered} \tag{2.2}$$

If the objective function or the constraint functions are numerically noisy, gradient information can be inaccurate, causing gradient based algorithms to fail by either diverging or converging to a false minimum. One solution is to fit a surrogate model to data derived from the design simulation, and use this smooth response surface in the optimization. Another approach is to use a gradient-free algorithm. Gradient free algorithms also have the advantage of allowing discrete variables in the design vector. Some perform a systematic, deterministic search of the design space, such

[1] L is the so-called Lagrangian function, the minimization of which satisfies the necessary conditions for the original NLP problem. λ and μ are the Lagrange multipliers, indicative of constraint sensitivity.

as DIRECT [22]. Other heuristic algorithms include Simulated Annealing (SA) [25] and the evolutionary Genetic Algorithm (GA) [7, 21]. While gradient free methods work when other algorithms do not, they also have the disadvantages of excessive function evaluation requirements and lack of optimality conditions. Except in special cases, a solution cannot be proven to be an optimum when a gradient free algorithm is employed.

Some algorithms are generally robust in that they reliably find the solution, while others (such as genetic algorithms) require tuning of parameters for each specific problem in order to be successful. An in-depth understanding of optimization theory and algorithms is essential to correctly pose and solve an NLP problem. Optimization software is not a push-button device that automates the design process. Haphazard application of optimization tools can produce erroneous results [32]. The reader should refer to one of the many texts on optimization texts if more background in optimization theory and algorithms is needed. Examples include: [6, 7, 32, 41].

2.1.2 Multi-Objective Optimization

The NLP problem posed in equation 2.1 aims to minimize a scalar objective function $f(\mathbf{X})$. Often multiple objectives are deemed important to a design, and these objectives are frequently conflicting. For example, mechanical designs commonly must minimize weight and compliance simultaneously, obviously conflicting objectives. The task of *multi-objective* optimization is regularly encountered in systems optimization, since it is usual for each subspace to have its own objective function. In automotive design, the system designer must maximize fuel economy, safety, comfort, road handling, acceleration, and reliability all simultaneously. Also, an important distinguishing factor between MDO formulations is how the multiple objective func-

tions are handled. The question then remains— how to minimize a vector valued function?

Strictly speaking, a vector cannot be minimized. A transformation must be made to repose the problem as a scalar optimization problem. The most ideal solution is to have a master function that takes all of the objective functions (and possibly other quantities) as inputs, and returns a value that truly represents the overall design objective. For example, this master function could calculate the expected profit of a product, or perhaps the cost per passenger mile of an aircraft. Unfortunately, comprehensive and accurate 'master functions' are not always available. Some other approach must be used.

An approximation to a master function can be made with a linear combination of the objective functions. If all of the objective functions are first scaled properly, a weight can be given to each objective function, indicating its relative importance. Normally these weights sum to one.

An alternative to the scalarized objective function above is to select one objective that is most important, and convert the remaining objectives to constraints, specifying bounds that they cannot violate. These bounding constraints are almost always active, yet were arbitrarily chosen by the designer, indicating that further study of the design problem is necessary before final conclusions are made.

In either the scalarized objective function or bounded objectives approach, the weights or the bounds respectively may be varied to explore the efficient frontier in the objective function space, also called the set of Pareto optimal designs [7, 32]. At a Pareto optimal design point no objective function can be improved without degrading the value of another objective. Figure 2.1 illustrates this idea with a two-objective function example. Both f_1 and f_2 are to be minimized. At the top left is

the best possible value of f_1, and at the bottom right is the best value of f_2. The intersection of these best possible values is the *utopia point*, which normally cannot be achieved. The curve in this plot is the Pareto set (or front); no design to the lower left of this front is possible. At any point on this front, to improve one objective, the other must be degraded.

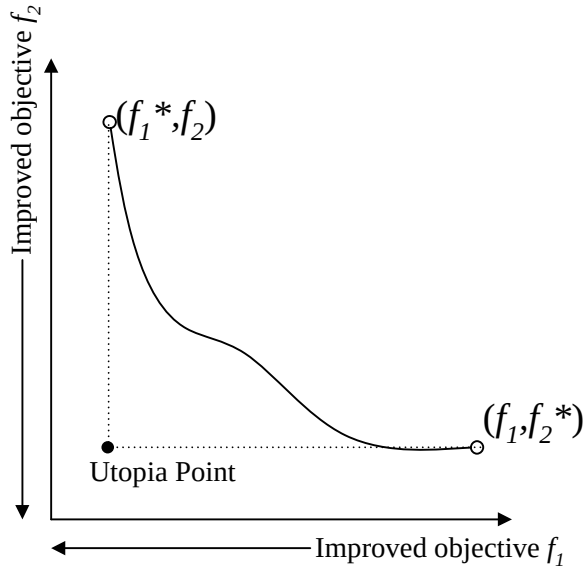

Figure 2.1: Objective function space of a two-objective function optimization.

2.1.3 Optimization of Partitioned Systems

The discussion of multi-objective optimization eluded to approaches for simultaneously optimizing the designs of several subsystems that together comprise a system. The ideal system objective function considers the impact each subsystem has on the entire system, and gives merit to what is truly sought from the system. Each subsystem needs to be a team player, sacrificing its own objectives if required to achieve the system optimal design. In other words:

A conglomeration of optimal parts does not comprise an optimal whole.

Let's begin this discussion of system optimization by examining the age-old, over-the-wall sequential design approach. This was first introduced in Section 1.2.2 with a hypothetical automotive design. It is termed over-the-wall since a design group that currently has the project works on its design without any real regard to other groups, and then passes it on when their part is done. This can pose serious problems. For example, if a final design is passed on to manufacturing without any prior communication with the manufacturing department, it may be discovered that the company does not have the facilities to build the product. Such problems have given rise to the push for parallelization and the study of concurrent engineering. All aspects of a product should be considered concurrently in order to produce a *system optimal design*, i.e. a product that has been optimized with respect to the overall product objective. If interactions between groups are not considered early on, and groups optimize their parts sequentially, side effects are generated that must be absorbed by subsequent groups, endangering overall system performance.

The sequential design strategy has an analogy in the discipline of Design of Experiments. In sequential design one *factor* in effect is being varied at a time, with all other factors being held constant. The factor in this sequential design analogy is the set of design variables a design group has control over. The shortcomings of the sequential, or *one-factor-at-a-time* (OFAT), approach are well known [45]. The OFAT approach is unlikely to lead to the true system optimum, and in practice requires far more function evaluations than other approaches. To illustrate, a two factor, single objective function example is used.

Objective function:

$$f(x_1, x_2) = a(x_1 - b)^2 + c(x_2 - d)^2 + ex_1x_2$$

The parameters $a - e$ were chosen to be $\{1,\ 1,\ 1,\ 1,\ -1\}^T$. Since $e \neq 0$, interaction is present in the system. The two factors x_1 and x_2 must work together to realize system optimality. What happens if an attempt is made to minimize $f(x_1, x_2)$ in a one-shot sequential approach? Suppose a starting point of $\mathbf{X}^0 = \{x_1\ x_2\}^T = \{3\ 5\}^T$ is given. Performing a line search in the x_1 direction, the partial derivative $\partial f/\partial x_1 = 0$ yields the point $\mathbf{X}^1 = \{3.5\ 5\}^T$. Moving to the next factor and performing a line search in the x_2 direction, the partial derivative $\partial f/\partial x_2 = 0$ yields the point[2] $\mathbf{X}^2 = \mathbf{X}^\ddagger = \{3.5\ 2.75\}^T$. This final design has a system response of $f(\mathbf{X}^\ddagger) = -0.3125$. This is substantially worse than the true system optimum $\mathbf{X}_* = \{2\ 2\}^T$, $f(\mathbf{X}_*) = -2.0$. The level sets of the objective function, the points generated by the OFAT procedure, and the true optimum are all illustrated in Figure 2.2.

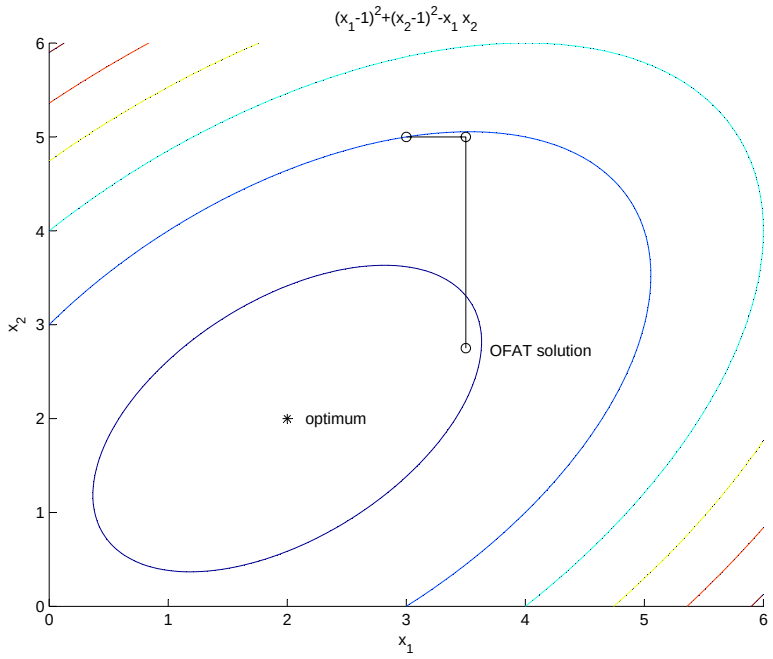

Figure 2.2: Sequence of points generated by the OFAT procedure.

[2] $\mathbf{X}^\ddagger$ denotes the pseudo-optimum found by the OFAT approach.

If iterations were allowed, the OFAT approach in this case could find the true optimum. However, many more function evaluations would be required than for approaches that consider the affect of both approaches simultaneously. Systems optimization commands consideration of all system members. The next section explains one method for analyzing a system as a whole, and much of the rest of this thesis is devoted to understanding ways to find system optima.

2.1.4 Complex System Analysis Terminology

Basic terminology for partitioned system analysis and optimization is introduced here. Consider the general coupled system illustrated in Figure 2.3. The system is partitioned into N subspaces, and all $N(N-1)$ possible subspace interactions are shown. In a real system all possible interactions, or *couplings*, do not necessarily exist.

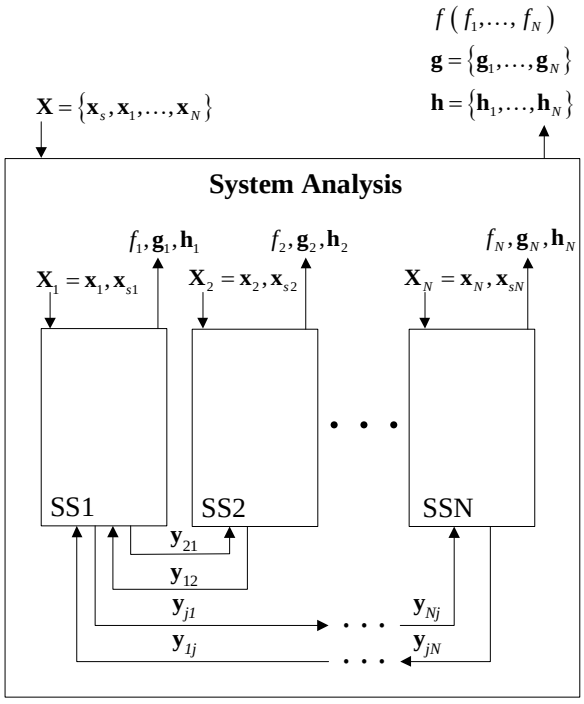

Figure 2.3: Diagram of a general non-hierarchic coupled system.

From an overall system perspective, the components within the system analysis box are considered to be a single monolithic entity. A designer, or an optimization algorithm, provides the system analysis with the design variable vector $\mathbf{X}$. When the process is complete, the system response functions, such as f, $\mathbf{g}$, and $\mathbf{h}$, are returned. With these system inputs and outputs defined, the system design problem fits the standard design problem formulation of equation 2.1.

From the definition of complex systems, it may not be possible to approach the system analysis as a single task. Partitioning the system into smaller subspaces ameliorates this problem. A clear notation system facilitates discussion of partitioned systems.

The system design variable vector is separated into two categories of variables: local design variables $\mathbf{x}_i$ and shared design variables $\mathbf{x}_{si}$. Local variables $\mathbf{x}_i$ are unique to subspace i— no other subspace takes them as inputs. Shared variables $\mathbf{x}_{si}$ are required inputs to subspace i, but are also used as inputs in at least one other subspace. The local variable vectors $\mathbf{x}_i$ have no common components, but the shared variable vectors $\mathbf{x}_{si}$ do. The collection of all local variables is $\mathbf{x}$, and the collection of all shared variables is $\mathbf{x}_s$. The aggregation of local and shared variables for subspace i is $\mathbf{X}_i$.

$$\mathbf{X} = \{\mathbf{x}_i^T, \mathbf{x}_s^T\}^T \quad i = 1 \ldots N$$

$$\mathbf{X}_i = \{\mathbf{x}_i^T, \mathbf{x}_{si}^T\}^T \quad \forall\, i$$

Each subspace may depend on outputs from one or more subspaces. The existence of these dependencies indicates the presence of interaction between subspaces. The collection of values generated by subspace j and received by subspace i is $\mathbf{y}_{ij}$. These quantities passed between subspaces are called coupling variables, and may be vector valued. The collection of all coupling variables input to system 1 for example is $\mathbf{y}_{1j}$,

where $j = 1 \ldots N$. The collection of all coupling variables is $\mathbf{y}$. Coupling variables are artifacts of decomposition; they do not exist in the system's original design problem statement. The set of coupling variables and the set of design variables have no common members.

Each subspace may generate a variety of responses other than coupling variables, such as subspace objective functions and constraint functions: f_i, $\mathbf{g}_i$, and $\mathbf{h}_i$. The system objective function f_s (sometimes shortened to f when the context is clear) may simply be one of the subspace responses, or in the general case is a function of several of the subspace responses. The set of all subspace constraints and any system level constraints is $\mathbf{g}$ and $\mathbf{h}$. As with the system objective function, system constraints may be a function of one or more subspace responses.

2.2 Analysis of Coupled Systems

Fundamental to the understanding of complex system optimization is the analysis of coupled systems. Some industries have dealt with the complexity of coupled system analysis for decades. For example, work done by the aerospace industry in this area is termed Multidisciplinary Analysis (MDA). Coupled systems may be viewed as simultaneous systems of nonlinear equations. Several methods exist to solve such systems when they cannot be solved explicitly, including iterative methods such as the *Newton-Raphson* method and *Fixed Point Iteration* [11]. The latter will be addressed in detail in this section.

Fixed Point Iteration (FPI) is regularly employed in the most basic MDO formulation, MDF, presented in Section 3.1. An exploration of the nature of FPI, including its limitations, illustrates the appropriate application of MDF, and gives impetus to the development of more sophisticated MDO strategies. Due to its intuitive imple-

mentation, MDF is the most frequently utilized MDO strategy [10]. Unfortunately, it is sometimes applied blindly without recognition of its sometimes significant shortcomings. This section, along with Chapters 3 and 4, develop the ideas pertinent to proper application of MDF and its associated analysis tool— FPI.

In this section the Fixed Point Iteration algorithm is presented, its convergence conditions will be proven in the two-dimensional case, and a hypothesis for convergence conditions in the n-dimensional case posed. Several examples will be employed to illustrate important concepts.

2.2.1 Fixed Point Iteration Algorithm

A simple coupled system with two subspaces is depicted in Figure 2.4. When the coupling variables[3] y_{ij} are written in a functional form such as $y_{21}(y_{12})$, this indicates that y_{21} is a dependent variable, and y_{12} is an independent variable[4]. The coupling variables can swap roles depending on what subspace is being considered. In addition, inverse functions may be found such that functions may be redefined in terms of a new independent variable, i.e. $y_{21}(y_{12})$ may be posed as $y_{21}(y_{21})$. The need for this notation will become clear shortly.

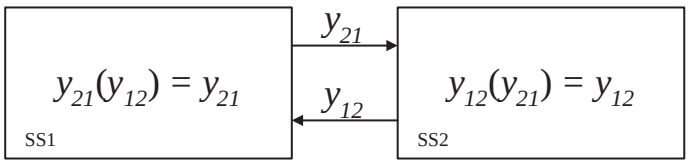

Figure 2.4: Two-dimensional coupled system.

This system possesses feedback coupling, since y_{12} depends on y_{21} and visa-versa.

[3] y_{ij} indicates a coupling variable that takes an output from SSj and sends it as an input to SSi.

[4] Only the variables pertinent to system analysis are shown in Figure 2.4. The coupling variables may indeed be dependent upon other quantities such as design variables. However, these additional quantities are held constant during analysis using FPI, and are omitted from the current presentation for clarity.

If one of the couplings did not exist, say y_{12}, then the system would have only feed-forward coupling, and analysis could be completed with a single sequential execution of $SS1$ and $SS2$. However, when feedback coupling exists, analysis is not so straightforward. If Fixed Point Iteration is employed, an initial guess is made for the input to the subspace executed first. In the example in Figure 2.4, a guess could be made for y_{12}, and then $SS1$ could be evaluated to obtain a value for y_{21}. This value is then used in the execution of $SS2$ to obtain an updated value of y_{12}. This output will not agree with the initial guess unless the guess was made at a so called *fixed point*. The resulting value of y_{12} can then be used as an updated guess for the input to $SS1$. If the system meets certain criteria, it will converge to a fixed point. The FPI algorithm for the two-dimensional example problem is:

(Step 0) choose initial guess y_{12}^0, set $i = 0$

(Step 1) $i = i + 1$

(Step 2) $y_{21}^i = y_{21}(y_{12}^{i-1})$

(Step 3) $y_{12}^i = y_{12}(y_{21}^i)$

(Step 4) if $|y_{12}^i - y_{12}^{i-1}| < \varepsilon$ stop, otherwise go to **(Step 1)**

The superscript i indicates the iteration number, and ε is the maximum inconsistency allowed between subspaces. Note that Step 3 uses the most recently updated input value, y_{21}^i; if y_{21}^{i-1} had been used instead the procedure would be termed Jacobi Iteration. When the inconsistency is less than ε, the system is said to be consistent. In other words, the output of the system coupling variables is equal to the guess for coupling variable values when a system is consistent. A point that produces this condition is called a fixed point $\mathbf{y_p}$, since further iterations will not change the location of the point. A fixed point is also termed an analysis solution. An n dimensional fixed point is expressed in equation 2.3.

$$\mathbf{y_p} = \mathbf{y}(\mathbf{y_p}) \tag{2.3}$$

Figure 2.5 exhibits graphically the convergence behavior of FPI. Case I shows oscillatory convergence, and Case II shows monotonic convergence. The initial guess is y_{12}^0, intermediate points are indicated with closed circles, and the fixed point is indicated with an open circle.

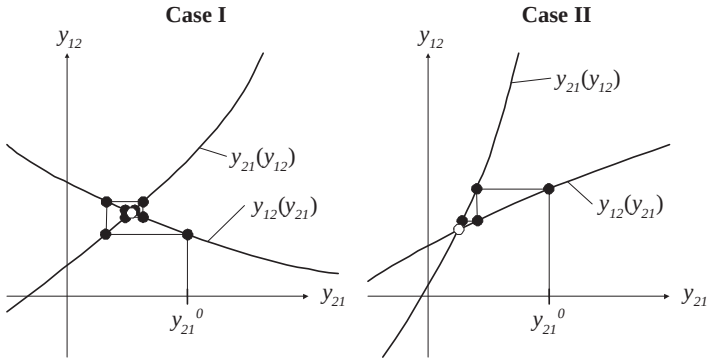

Figure 2.5: Oscillatory and monotonic FPI convergence.

The order of function evaluation and the function derivatives affect both the convergence rate and the possibility of convergence. In general FPI convergence is linear (when it occurs). Convergence may be quadratic in certain cases, such as when $y_{12}(y_{21})$ has a slope (with respect to y_{21}) near zero. The Newton-Raphson method has quadratic convergence in general, but is used less frequently in practice because of the required derivative information.

In the two convergent examples shown in Figure 2.5 it is clear that the equivalent relationships expressed in equation 2.4 hold in the neighborhood of a fixed point $\mathbf{y_p}$.[5] What happens to FPI convergence if these conditions do not hold?

[5]Observe that inverse functions must be computed in order to evaluate the expressions in equation 2.4.

$$\left|\frac{\partial y_{21}(y_{21})}{\partial y_{21}}\right| > \left|\frac{\partial y_{12}(y_{21})}{\partial y_{21}}\right| \Leftrightarrow \left|\frac{\partial y_{12}(y_{12})}{\partial y_{12}}\right| > \left|\frac{\partial y_{21}(y_{12})}{\partial y_{12}}\right| \tag{2.4}$$

Figure 2.6 illustrates one such case where equation 2.4 does not hold. A fixed point exists, yet the algorithm diverges from this fixed point. FPI cannot find the fixed point, and is termed a repelling fixed point. If equation 2.4 does hold in the neighborhood of a fixed point $\mathbf{y_p}$, then FPI will converge, and $\mathbf{y_p}$ is termed an attractive fixed point. If equation 2.4 holds with equality, then two possibilities exist. First, the functions are parallel and no fixed point exists if the equality holds for the entire analysis domain. In this case FPI will march off toward infinity[6]. Second, the functions are orthogonal and are oriented such that FPI will oscillate infinitely through a set of points, i.e. the system has oscillatory divergence.

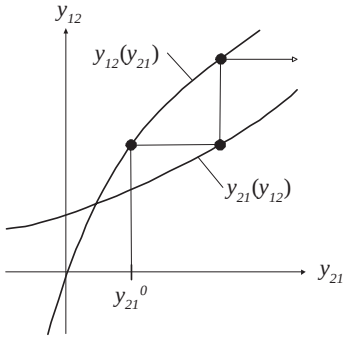

Figure 2.6: Divergent FPI behavior.

Some systems have multiple fixed points, and FPI may not be capable of finding them all if the system has any repelling fixed points. The point FPI actually converges to depends upon the location of the initial guess (starting point). An example of a system with multiple fixed points is shown in Figure 2.7.

Closed circles indicate attractive fixed points; open circles indicate repelling fixed points. A starting point near to one of the attractive fixed points will converge to

[6]The case may also exist where no fixed point exists even if equation 2.4 holds due to some degeneracy. This discussion is limited to systems with existent fixed points

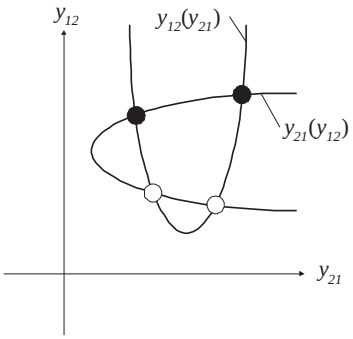

Figure 2.7: System with multiple fixed points. Closed circles are attractive fixed points; open circles are repelling fixed points.

that point. FPI will never find either of the repelling fixed points. If FPI is used as an analysis tool for optimization, it will never be known if one of the repelling fixed points would have led to a better solution. For example, if the objective of the original design problem was to minimize $y_{12} + y_{21}$, then the lower left fixed point is the obvious best solution. However, FPI cannot find this point, and may result in either of the top two points being presented as an 'optimal' solution. This significant shortcoming lends additional motivation to more sophisticated formulations that use optimization algorithms to drive the analysis, rather than FPI. Such formulations drive the process toward the best analysis solution.

FPI may easily be extended to n dimensions. Additional guesses may be required at multiple points within the algorithm. An example problem with more than two subspaces utilizing FPI for analysis will be developed in Chapter 6.

2.2.2 Convergence of Fixed Point Iteration

Informal statements concerning Fixed Point Iteration were given in Section 2.2.1. This section more rigorously develops FPI convergence conditions. If the system can be written in the scalar form $x = g(x)$, necessary and sufficient conditions for convergence can easily be proven. It can also be shown that these conditions are

equivalent to equation 2.4. In this section well known conditions for the n dimensional case are presented, and a less strict set of convergence conditions is presented as a hypothesis.

The coupled system in Figure 2.4 can be written as a single composite function (equation 2.5).

$$y_{12} = y_{12}(y_{21}(y_{12})) = y_{12} \circ y_{21}(y_{12}) \tag{2.5}$$

Here the only independent variable is y_{12}, and the composite function $y_{12} \circ y_{21}(y_{12})$ fills the role of $g(x)$ described above. The FPI algorithm reduces to a recursion formula $y_{12}^{i+1} = g(y_{12}^{i})$. Assuming a fixed point $y_{12p} = g(y_{12p})$ exists, we can subtract the recursion formula from the fixed point equation to obtain:

$$y_{12p} - y_{12}^{i+1} = g(y_{12p}) - g(y_{12}^{i})$$

Appealing to the derivative mean value theorem, $\exists\ \xi$ such that:

$$g'(\xi) = \frac{g(y_{12p}) - g(y_{12}^{i})}{y_{12p} - y_{12}^{i}}$$

If g is C_1 continuous on the interval $[y_{12p}\ y_{12}^{i}]$, then:

$$y_{12p} - y_{12}^{i+1} = g(y_{12p}) - g(y_{12}^{i}) = g'(\xi)(y_{12p} - y_{12}^{i})$$

The true error $\epsilon_i = y_{12p} - y_{12}^{i}$ decreases if and only if $g'(\xi) < 1$, since:

$$\epsilon_{i+1} = g'(\xi)\epsilon_i$$

Therefore, if a fixed point y_{12p} exists, the condition that $g'(y_{12}) < 1\ \forall$ points in the neighborhood of y_{12p} is necessary and sufficient for convergence of FPI if the

algorithm is given a starting point in the neighborhood of y_{12p}. Using the chain rule, this condition can be put in terms of the original problem (equation 2.6).

$$\left|\frac{\partial}{\partial y_{12}} g(y_{12})\right| = \left|\frac{\partial}{\partial y_{12}} y_{12}(y_{21}(y_{12}))\right| = \left|\frac{\partial y_{12}(y_{21})}{\partial y_{21}} \frac{\partial y_{21}(y_{12})}{\partial y_{12}}\right| < 1 \tag{2.6}$$

The composite function may alternatively be written with the reverse evaluation order as $g = y_{21}(y_{12}(y_{21}))$. In this case the convergence condition takes the form of equation 2.7.

$$\left|\frac{\partial y_{21}(y_{12})}{\partial y_{12}} \frac{\partial y_{12}(y_{21})}{\partial y_{21}}\right| < 1 \tag{2.7}$$

The conversion from a two dimensional from to a composite function form can be viewed graphically (Figure 2.8). The plot on the left is a linear system before conversion, and the plot on the right is after conversion. The system in this example is:

$$y_{21}(y_{12}) = \frac{1}{4} y_{12} - \frac{1}{4}$$

$$y_{12}(y_{21}) = 2 + 2y_{21}$$

The fixed point may be explicitly found, and is $\{0.5, 3.0\}^T$. The composite function is:

$$g(y_{21}) = y_{21}(y_{12}(y_{21})) = \frac{1}{4} + \frac{1}{2} y_{21}$$

The solution of the fixed point formula $y_{21} = g(y_{21})$ is $y_{21} = 0.5$ and $y_{12} = 3.0$, the same as the original system.

As expected, the convergence properties are unaffected by the conversion. The condition in equation 2.4 is satisfied in both cases. In the plot on the right the proven

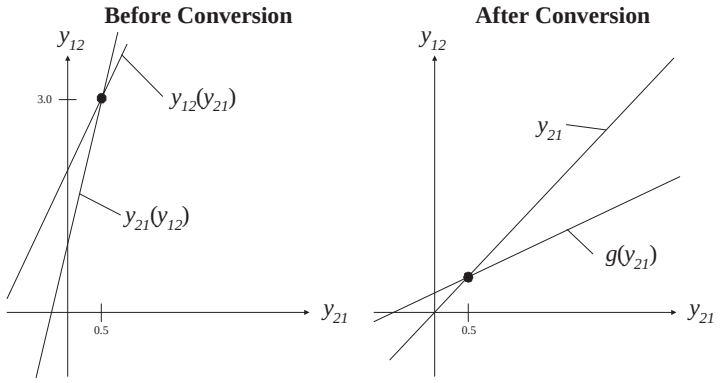

Figure 2.8: Illustration of conversion from a two-equation system to composite function $x = g(x)$ form.

convergence condition $|g'(y_{21})| < 1$ is satisfied. Equation 2.4 is as yet unproven, only informally presented. It can be rigorously shown that $|g'(y_{21})| < 1$ and equation 2.4 are equivalent. First observe that:

$$|g'(y_{12})| = \left| \frac{\partial}{\partial y_{12}} g(y_{12}) \right| = \left| \frac{\partial y_{12}(y_{21})}{\partial y_{21}} \frac{\partial y_{21}(y_{12})}{\partial y_{12}} \right|$$

Then beginning with the first instance of equation 2.4, and using the above expression, the proof is as follows:

$$\left| \frac{\partial y_{21}(y_{21})}{\partial y_{21}} \right| > \left| \frac{\partial y_{12}(y_{21})}{\partial y_{21}} \right|$$

$$\left| \frac{\partial y_{21}(y_{21})/\partial y_{21}}{\partial y_{12}(y_{21})/\partial y_{21}} \right| > 1$$

$$\left| \frac{\partial y_{12}(y_{21})}{\partial y_{21}} \left(\frac{\partial y_{21}(y_{21})}{\partial y_{21}} \right)^{-1} \right| < 1$$

and since when finding the appropriate inverse function $\left(\frac{\partial y_{21}(y_{21})}{\partial y_{21}} \right)^{-1} = \frac{\partial y_{21}(y_{12})}{\partial y_{12}}$ we see that

$$\left| \frac{\partial y_{21}(y_{12})}{\partial y_{12}} \frac{\partial y_{12}(y_{21})}{\partial y_{21}} \right| = |g'(y_{12})| < 1$$

Therefore, if a fixed point y_{12p} exists, and if the condition given in equation 2.4 holds $\forall$ points in the neighborhood of y_{12p}, necessary and sufficient conditions for convergence of FPI are met if the algorithm is given a starting point in the neighborhood of y_{12p}. The process for proving the second instance of equation 2.4 may be either as above, or simply to show that the two instances are equivalent, which was stated without proof in §2.2.1. The intuitive observations from Figure 2.5 are indeed correct.

2.2.3 Coupled Systems in n Dimensions

The concepts developed in Sections 2.2.1 and 2.2.2 may be extended to higher dimension systems. If n subspaces exist, then $n(n-1)$ possible couplings exist. Each of these couplings may also be vector valued. A convenient representation of such a system is the Design Structure Matrix (DSM) [15], a type of adjacency matrix. A DSM is shown in Figure 2.9 for a hypothetical system with four subspaces.

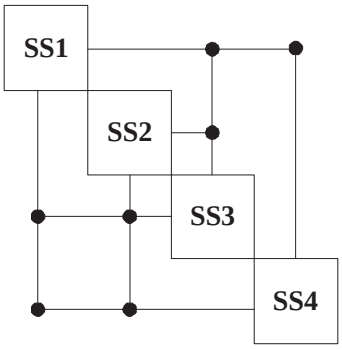

Figure 2.9: Design Structure Matrix of a hypothetical four-subspace system.

Each square represents a subspace. When in the context of FPI the evaluation sequence begins at the top left subspace and progresses to the lower right subspace.

Within the upper right diagonal of the matrix lines emanating from the right side of subspaces are outputs, and the lines entering the top of a subspace are inputs. The lines in the upper diagonal represent feedforward couplings. Within the lower left diagonal lines emanating from the left side of subspaces are outputs, and lines entering the bottom of subspaces are inputs. The lines in the lower diagonal of the DSM are feedback couplings. Nodes indicate where a connection is made. For example, $SS3$ feeds back to $SS1$, but not to $SS2$.

Existence of feedback couplings requires iterations to arrive at an analysis solution. If no feedback couplings existed, then a single sequence of subspace evaluation would complete the analysis. For systems with feedback coupling the subspace evaluation sequence may be adjusted to minimize the number of feedback couplings (a discrete optimization problem). This task is more involved than the above description intimates. The DSM representation hides several complexities. Each of the couplings may be vector valued— it may be more valuable to eliminate a single large vector feedback coupling than any combination of the others. In addition, the coupling strength (defined shortly) may vary from coupling to coupling, as well as vary within both the analysis space $\mathbf{y}$ and in the design space $\mathbf{x}$ (when analysis is used in an optimization context). Finally, changing the evaluation sequence may alter the subspace derivative relationships in such a way to render to system non-convergent. All of these points indicate that determining the merit of a particular sequence is not as straightforward as counting the number of feedback couplings.

Couplings that cannot be eliminated through reordering the evaluation sequence may still be temporarily eliminated. Criteria for determining when in the FPI algorithm these suspensions should be made have been developed [15]. In some cases the additional computational expense of evaluating this suspension criteria outweighs

computational gains realized through the suspensions. As with many strategies discussed throughout this thesis, choice of whether to apply them or not is highly problem dependent.

The notion of coupling strength was used in the above discussion. Coupling strength may be viewed as how sensitive the output of one subspace is to its input from other subspaces. For example, in the turbine blade example introduced in Chapter 1, if the modulus of elasticity (E) is low, then within the structural analysis subspace the blade will elongate more due to rotation. This exposes more surface area to hot gasses, resulting in a higher temperature distribution and greater thermal expansion. The longer blade is then subject to higher inertial forces, impacting the structural analysis again. In other words, with low E, each iteration of the feedback loop possesses more change than compared to a system with a high E. Similar arguments may be made if the thermal expansion α is high. A turbine blade analysis with a low E or high α will require many iterations to converge. The sensitivity between the subspaces depends on their relative derivatives, which are also important to convergence conditions. If the conditions in equation 2.4 are satisfied, but are near equality, then convergence will occur, but very slowly. Convergence speed of FPI may be considered another measure of coupling strength. Note that the value of the derivatives in equation 2.4 in general vary over both the analysis space $\mathbf{y}$ and in the design space $\mathbf{x}$. Characterizing the coupling nature of a system is a complex task indeed. Coupling strength may also be viewed as the strength of interaction between subspaces in a regression sense.

A well-known sufficiency condition for n dimensional coupled systems is given in equation 2.8[7].

[7]Functional notation has been omitted for conciseness. Functional dependencies should be clear from context.

$$\begin{aligned} \left|\frac{\partial y_1}{\partial y_1}\right| + \ldots + \left|\frac{\partial y_1}{\partial y_n}\right| &< 1 \\ \vdots & \quad \vdots \\ \left|\frac{\partial y_n}{\partial y_1}\right| + \ldots + \left|\frac{\partial y_n}{\partial y_n}\right| &< 1 \end{aligned} \tag{2.8}$$

A fixed point in the general case was defined by equation 2.3 ($\mathbf{y_p} = \mathbf{y}(\mathbf{y_p})$). If the starting point is sufficiently close to $\mathbf{y_p}$, then FPI will converge if equation 2.8 holds $\forall$ points in the neighborhood of $\mathbf{y_p}$. Please note that this condition is sufficient and not necessary. It is not hard to find examples that violate equation 2.8, yet are still convergent. The example from Figure 2.8 is one such case.

It is interesting to consider the possibility of looser sufficiency conditions that include more of the set of convergent systems, or perhaps even necessary and sufficiency conditions that include the entire set. By extending the technique utilized to informally develop equation 2.4, a similar condition for a three-dimensional system has been generated. By observing the projections of the three system analysis functions, y_1, y_2, and y_3, on to each of the three planes in three-space, intuitively the conditions presented in equation 2.9 are the three dimensional version of equation 2.4. An abbreviated notation similar to that used in equation 2.8 is adopted for these conditions, i.e. the functional notation is dropped, and a single subscript is used to denote what subspace evaluates the function. The appropriate functional dependencies can be inferred from the projection plane indicated. These conditions are presented as an hypothesis without proof. It is proposed that FPI in three dimensions will converge if and only if equation 2.9 holds in the neighborhood of an existent fixed point $\mathbf{y_p}$, and the starting point for FPI is sufficiently close to $\mathbf{y_p}$. By decomposing an n dimensional problem into appropriate projections, these conditions can

be extended to n dimensions.

$$\begin{aligned}
\left|\frac{\partial y_1}{\partial y_1}\right| > \left|\frac{\partial y_3}{\partial y_1}\right| \quad &\Leftrightarrow \quad \left|\frac{\partial y_3}{\partial y_3}\right| > \left|\frac{\partial y_1}{\partial y_3}\right| \text{ (in the } y_1\text{–}y_3 \text{ plane)} \\
\left|\frac{\partial y_1}{\partial y_1}\right| > \left|\frac{\partial y_2}{\partial y_1}\right| \quad &\Leftrightarrow \quad \left|\frac{\partial y_2}{\partial y_2}\right| > \left|\frac{\partial y_1}{\partial y_2}\right| \text{ (in the } y_1\text{–}y_2 \text{ plane)} \\
\left|\frac{\partial y_2}{\partial y_2}\right| > \left|\frac{\partial y_1}{\partial y_2}\right| \quad &\Leftrightarrow \quad \left|\frac{\partial y_2}{\partial y_1}\right| > \left|\frac{\partial y_1}{\partial y_1}\right| \text{ (in the } y_2\text{–}y_3 \text{ plane)}
\end{aligned} \tag{2.9}$$

The nature of FPI has been explored, and several advantages and shortcomings have been identified. FPI is frequently the strategy of choice to analyze coupled systems because it is intuitive to implement, requires no derivative information, and usually no modification of subspace analysis tools. However, FPI does not always produce an analysis answer [2, 28], even when one or more analysis solutions (fixed points) exist. In addition, FPI is sequential in nature, which extends analysis time. If analysis of coupled systems is being used for optimization, one alternative is to use the optimization algorithm to drive the analysis procedure (i.e. driving the subspaces into agreement). With the optimizer in control, solutions may be found that could not with FPI, and if multiple analysis solutions exist the optimizer is driven to find the best solution.

CHAPTER 3

Fundamental MDO Strategies

This chapter introduces the fundamental strategies for MDO, and then outlines a framework that can be used to categorize MDO formulations in general. The first three formulations are considered *single-level approaches* because an optimizer is only employed at one level— the system level. The analysis is distributed via partitioning described in Chapters 1 and 2, yet design is not distributed. All decisions are made by the single system-level optimizer. The three formulations discussed in this chapter were formally introduced in the seminal publication: *Problem Formulation for Multidisciplinary Optimization* [13].

In many cases single-level approaches perform well. However, several reasons exist for extending MDO to *multi-level approaches*. The complete design centralization of single-level approaches may not map well to existing organizational structures. In addition, this centralization may demand excessive data communication requirements. Centralization also excludes the utilization of disciplinary experts or specialized optimization procedures. Multi-level approaches allows for utilization of local decision making resources, and can reduce communication requirements. The discussion of the general MDO framework in this chapter introduces the structure of multi-level approaches, and Chapter 5 develops in detail two selected multi-level formulations.

3.1 MDF

The most basic of MDO formulations is the *Multidisciplinary Feasible* (MDF) approach, also known as *Nested Analysis And Design* (NAND), *Single-NAND-NAND* (SNN), *All-in-One*, *One-at-a-Time*, and *All-at-Once* (AAO). Though all of these names fit the approach, the number of names for this formulation is a source of confusion, particularly when it is called AAO (a name reserved for a very different single-level formulation presented in §3.3).

The MDF architecture (2-subspace example) is depicted in Figure 3.1. A single system-level optimizer is used, and from the perspective of the optimizer MDF is no different than a general design problem. A system analyzer coordinates all of the subspace analyzers. The optimizer supplies the system analyzer with a design $\mathbf{x}$, and the system analyzer supplies the optimizer with the appropriate response functions, f, $\mathbf{g}$, and $\mathbf{h}$. MDF maintains the structure of non-hierarchical problems.

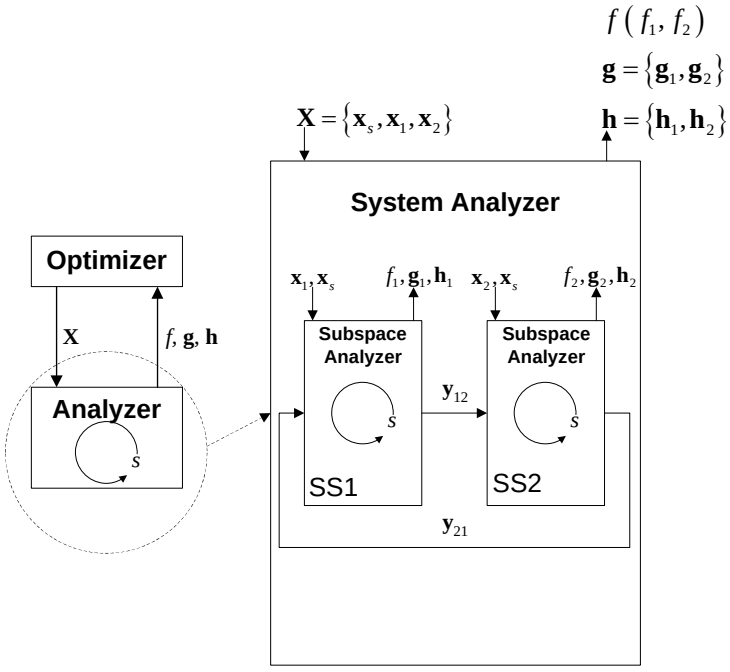

Figure 3.1: MDF structure.

When the system analyzer supplies the response, it has already converged to a consistent analysis solution. If the subsystems are coupled, the FPI algorithm[1] (§2.2.1) is typically used to find an analysis solution at a fixed design point $\mathbf{x}$. In other words, within each optimization iteration, a complete analysis solution is found. The analysis is nested within the design (hence the name NAND). The optimizer is charged with the responsibility to find the optimal design $\mathbf{x}_*$ (the design solution), while the system analyzer is solely responsible to find the set of consistent coupling variables $\mathbf{y_p}$ (i.e. a fixed point, or the analysis solution). In addition, the subspace analyzers are responsible to find state variables $\mathbf{s}$ that satisfy governing equations if applicable[2].

MDF refers to any complex system optimization strategy that performs a complete system analysis at every optimization iteration, regardless of analysis method. In industries that deal with coupled systems, Fixed Point Iteration is regularly employed as the analysis tool. However, other analysis tools may be used within an MDF approach. Section 7.1.2 provides one example of MDF that does not employ Fixed Point Iteration. If a design team employs whatever means necessary to arrive at a system analysis solution for every optimization iteration (or in other words for every design proposed by the optimizer), the team is using an MDF strategy. Similarly, if a monolithic computational analysis tool is used to calculate an analysis solution at every optimization iteration, regardless of whether FPI is employed or not, MDF is being employed.

The designation MDF stems from the property that at every optimization iteration, the system analysis is consistent. This terminology stems from the older field

[1]The Fixed Point Iteration algorithm of §2.2.1 is but one of several options for system analysis

[2]Many analyses find solutions to differential equations, such as elasticity or Navier-Stokes equations for solid mechanics or fluid analysis respectively. The state variables describe the state of the system, such as the strain field or velocity field.

of Multidisciplinary Analysis (MDA). One term used to describe a consistent system is *Multidisciplinary Feasibility* [13]. In other words, the multiple disciplines are feasible because all of the coupling variables match. Multidisciplinary feasibility may be thought of as analysis feasibility. The analysis is *feasible* at an analysis solution. This terminology has been a source of confusion. During the course of an MDF design process, the system may in fact possess multidisciplinary feasibility (system consistency), but lack design feasibility (i.e. $\mathbf{g}$ or $\mathbf{h}$ from the original design problem is violated). To avoid any obfuscation, in this thesis feasibility will refer only to design feasibility, and the terms system consistency and analysis solution will be used to refer to the consistent matching of coupling variables (i.e. a fixed point).

MDF is completely non-hierarchic in nature. There are no restrictions on data communication between the subspaces. In a purely computational context, this approach is desirable if the subspaces are weakly coupled (fast analysis convergence), and if the subspace analyses are not computationally expensive. In an organizational context, MDF allows the continued use of legacy analysis tools without modification (only formal communication requirements may need to be instituted) [13]. If the organization already performs a complete analysis before making a design decision, MDF is a natural fit.

The MDF formulation is shown in equation 3.1. Little difference exists between this formulation and the original design problem (equation 2.1). In the MDF formulation system decomposition is denoted by the variable and function partitioning. The set of constraint equations for the entire system $\mathbf{g}$ and $\mathbf{h}$ are the union of all the constraints for the subspaces, and possibly additional system level constraints. The system objective function may be one of the subspace objective functions, or a function of several of them.

$$\begin{aligned} \min_{\mathbf{X}=\{\mathbf{x},\mathbf{x_s}\}} \quad & f(\mathbf{X}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{X}) = \left\{\mathbf{g_1}^T, \mathbf{g_2}^T, \ldots, \mathbf{g_s}^T\right\}^T \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{X}) = \left\{\mathbf{h_1}^T, \mathbf{h_2}^T, \ldots, \mathbf{h_s}^T\right\}^T = \mathbf{0} \end{aligned} \tag{3.1}$$

It turns out that MDF is the most widely applied MDO formulation in many industries, including aerospace [10]. Much work has been done to demonstrate the usefulness of such a system design approach [15]. Optimizing the system as a whole generally produces significantly better results than optimizing subspaces individually without regard to system performance. Improvements in efficiency of MDF have been realized through such tactics as reordering the analysis sequence to minimize feedback coupling, and temporarily suspending couplings that are weak at a particular point in the design and analysis space (§2.2.3).

Some systems may be ordered such that there is no feedback coupling. In this case no iteration is required for analysis— a single sequential analysis is sufficient. MDF may be a natural choice for such a design problem, although other approaches may be beneficial by parallelizing the analysis.

The merits of MDF are notable, however it is important to clearly understand its limitations as well. The sometimes large iteration loops of MDF can be computationally expensive, particularly if the system is strongly coupled and the system analysis requires many iterations to converge. If a gradient-based optimization algorithm is employed, several more analyses must be performed for finite differencing. MDF is aptly considered a *brute force* approach [10]. When MDF uses FPI as an analysis tool, it is afflicted with all of the same problems as FPI. Section 2.2 was included primarily to illustrate the important shortcomings of FPI as it applies to its utilization in MDF. A review of these shortcomings is listed below.

- FPI is frequently non-convergent.
- FPI may not be capable of finding all fixed points.
- FPI is a sequential analysis tool.

MDF can exhibits low robustness as a design strategy. If the analysis does not converge for even one design point (or finite difference point), the optimizer may fail. If more than one analysis solution exists, MDF may fail to find the solution that leads to the optimal design[3]. Because MDF cannot be parallelized, design cycle time may be significant. One analysis may be sitting idle for significant periods of time waiting for required input. In an organization this can be extremely inefficient.

These limitations motivate the development of more sophisticated formulations that offer better convergence properties, fit a wider variety of organizational structures, and allow for parallelization of analysis. The rest of this thesis is devoted to understanding a selection of these additional formulations.

3.2 IDF

To address some of the limitations of the MDF formulation, the *Individual Disciplinary Feasible* approach was developed. IDF is also known as *Simultaneous Analysis And Design* (SAND) or *Single-SAND-NAND*. Like MDF, an analyzer for each subspace is employed (solving for state variables if required), and a single system-level optimizer is used. The key difference is that the optimizer coordinates the interactions between the subspaces, rather than relying on the simple iterative scheme of Fixed Point Iteration, or some other analysis tool. This enables parallelization, improves convergence properties, and drives the design toward better solutions if multiple analysis solutions exist.

[3]In this case MDF may indeed find a design solution that satisfies optimality conditions, but may not be capable of finding another existent design with better performance.

The IDF architecture is illustrated in Figure 3.2. The system optimizer gains additional responsibility for the solution process over the MDF approach. In addition to deciding the appropriate values for the design variables, the system optimizer must also control the values for the coupling variables $\mathbf{y}$. Rather than relying on simple iteration to determine the next coupling variable values, an optimization algorithm efficiently performs this task instead, improving convergence speed and probability of convergence. IDF has notably improved robustness over MDF. Note that the distinction between analysis and design processes is blurred— they are performed simultaneously (hence the name SAND). The system optimizer provides all of the inputs required for all subspaces. Since the subspaces no longer must wait for the conclusion of other analyses before commencing their own analysis, all of the subspaces may be evaluated in parallel. As with MDF, design decision making is centralized in the IDF formulation, and analysis is distributed.

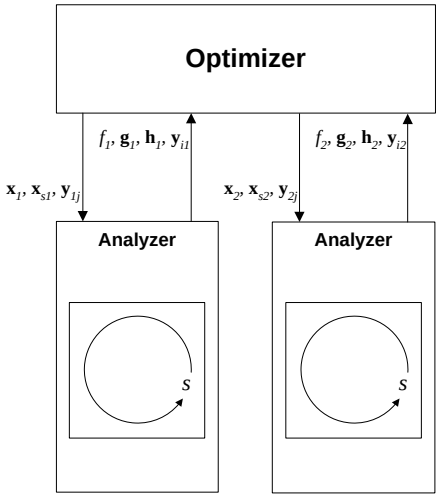

Figure 3.2: IDF architecture.

In order to ensure the system is consistent, auxiliary equality constraints in the system optimizer are required. These constraints require that the coupling variables computed by the subspaces $\mathbf{y}(\mathbf{X}, \mathbf{y})$ are equal to the coupling variables supplied by

the optimizer $\mathbf{y}$. This system consistency is only enforced at convergence of the system design problem. If for some reason the process must be interrupted, the intermediate design may not be consistent, in addition to the possibility of lacking design feasibility. In contrast, if an MDF process is interrupted prematurely, the system will be consistent (but may lack design feasibility).

The auxiliary constraints perform another important function— they break down any non-hierarchical links in the system so that the problem can be solved in a hierarchical manner, as exhibited in Figure 3.2. This principle applies in general, i.e., any arbitrary non-hierarchical system can be reposed as a hierarchical system through the use of auxiliary constraints [10]. Methods intended for strictly hierarchical systems may be used to solve non-hierarchical problems when auxiliary constraints are used to effect this transformation. Though this approach is possible, the transformed problem may not behave well.

Because the IDF approach does not ensure that multiple disciplines are consistent (i.e. Multidisciplinary Feasibility) at each optimization iteration, but rather ensures only that each discipline satisfies its governing equations, it is called the Individual Disciplinary Feasible approach. The degree of centralization in IDF is higher than in MDF. The dimension of the optimization problem is increased due to the coupling variables becoming decision variables, and the data communication requirements are higher. IDF maps to a design organization with a single project manager, making all of the design decisions and guiding the analysis groups into agreement.

The formulation of the IDF approach is given in equation 3.3. It is similar to the MDF formulation, except that now the optimization is performed with respect to both the design variables $\mathbf{X}$ *and* the coupling variables $\mathbf{y}$, and that the auxiliary constraints $\mathbf{h_{aux}}(\mathbf{X}, \mathbf{y})$ are included to ensure system consistency.

$$\begin{aligned} \min_{\mathbf{X}=\{\mathbf{x},\mathbf{x_s}\},\mathbf{y}} \quad & f(\mathbf{X},\mathbf{y}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{X},\mathbf{y}) = \left\{\mathbf{g_1}^T, \mathbf{g_2}^T, \ldots, \mathbf{g_s}^T\right\}^T \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{X},\mathbf{y}) = \left\{\mathbf{h_1}^T, \mathbf{h_2}^T, \ldots, \mathbf{h_s}^T\right\}^T = \mathbf{0}. \\ & \mathbf{h_{aux}}(\mathbf{X},\mathbf{y}) = \mathbf{y}(\mathbf{X},\mathbf{y}) - \mathbf{y} = \mathbf{0}. \end{aligned} \tag{3.2}$$

In both a computational and an organizational context the parallel nature of IDF has an advantage over the sequential MDF approach. If parallel analysis tools (multiple analysis groups or parallel processors) are available, IDF can offer a significant compression of the design process. If a high level of centralization is acceptable, then IDF may be an ideal design strategy.

3.3 AAO

The last of the three fundamental single-level MDO approaches covered in this thesis is the *All-At-Once* strategy (AAO). It is also referred to as *Single-SAND-SAND*, and sometimes just SAND. Occasionally the term AAO is erroneously used to refer to the MDF approach; it is important to make the distinction between the two formulations.

AAO is a highly centralized approach. Instead of utilizing analyzers to complete the analysis for each subspace, evaluators are used that compute only the residuals of the governing equations. The system optimizer is now saddled with three sets of decision variables: the original design variables $\mathbf{X}$, the coupling variables $\mathbf{y}$, and the state variables $\mathbf{s}$. AAO centralizes both design and analysis, but still distributes evaluation of governing equations. This high degree of centralization offers impressive efficiency in some situations, yet is sometimes difficult to map to many organizational structures due to its centralization and specialized structure. Figure 3.3 illustrates

how MDF and AAO may be viewed as opposite extrema with respect to the number of decision variables the system optimizer explicitly controls. IDF is an intermediate occupant of this spectrum.

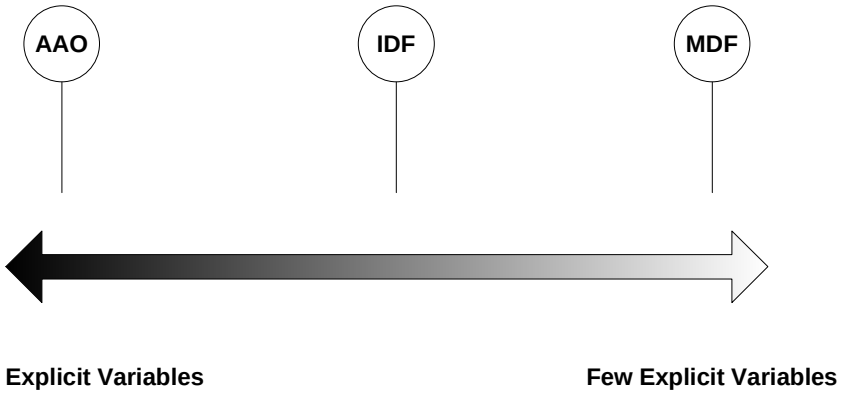

Figure 3.3: Classification of single-level formulations based on number of explicitly controlled decision variables.

The AAO architecture is illustrated in Figure 3.4. The state variable vector is divided into the state variables for each subspace ($\mathbf{s} = \left\{\mathbf{s_1}^T, \mathbf{s_2}^T, \ldots\right\}^T$), and the residuals[4] for each subspace $\mathbf{w_i}$ are reported to the optimizer along with other pertinent values.

The formulation of the AAO approach is given in equation 3.3. It is similar to the IDF formulation, but includes an additional auxiliary constraint to ensure zero residuals at problem convergence. The optimization is performed with respect to the design variables $\mathbf{X}$, the coupling variables $\mathbf{y}$, *and* the state variables $\mathbf{s}$. This approach is truly All-At-Once, since design, system analysis, and subspace analysis are all performed simultaneously.

[4]Residuals quantify how well the state variables satisfy the governing equations. Zero residuals indicate exact satisfaction.

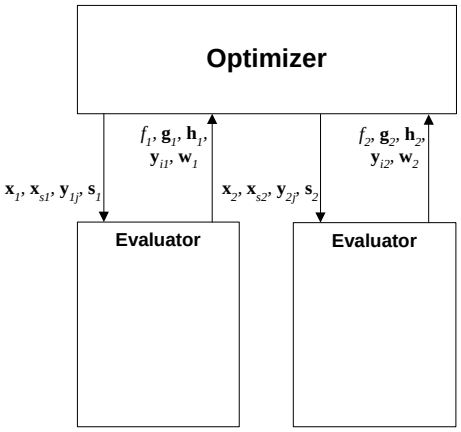

Figure 3.4: AAO structure.

$$
\begin{aligned}
\min_{\mathbf{X}=\{\mathbf{x},\mathbf{x_s}\},\mathbf{y},\mathbf{s}} \quad & f(\mathbf{X},\mathbf{y}) \\
\text{subject to} \quad & \mathbf{g}(\mathbf{X},\mathbf{y},\mathbf{s}) = \left\{\mathbf{g_1}^T, \mathbf{g_2}^T, \ldots, \mathbf{g_s}^T\right\}^T \leq \mathbf{0} \\
& \mathbf{h}(\mathbf{X},\mathbf{y},\mathbf{s}) = \left\{\mathbf{h_1}^T, \mathbf{h_2}^T, \ldots, \mathbf{h_s}^T\right\}^T = \mathbf{0}. \\
& \mathbf{h_{aux}}(\mathbf{X},\mathbf{y},\mathbf{s}) = \left\{ \begin{array}{c} \mathbf{y}(\mathbf{X},\mathbf{y},\mathbf{s}) - \mathbf{y} \\ \\ \mathbf{w}(\mathbf{X},\mathbf{y},\mathbf{s}) \end{array} \right\} = \mathbf{0}.
\end{aligned}
\tag{3.3}
$$

In [13] the three foregoing formulations are described in more detail, and are compared with respect to important performance indices. Selection of the appropriate formulation is facilitated by an understanding of the strengths and weaknesses of each. Table 3.1 is adapted from [13], and outlines several important characteristics of each formulation.

3.4 General MDO Framework

A general framework has been developed that is capable of characterizing most MDO formulations (Balling and Sobieszczanski-Sobieski [5]). This notation system has been widely used in the MDO community to concisely describe the structure of

Table 3.1: Comparison of single-level MDO formulation characteristics.

	AAO	**IDF**	**MDF**
▷Use of legacy analysis tools	None	Full	Full, requires coupling
▷Satisfaction of governing eqns.	Only at convergence	At each opt. iteration	At each opt. iteration
▷System consistency	Only at convergence	Only at convergence	At each opt. iteration
▷Optimizer decision variables	$\mathbf{X} = \{\mathbf{x}, \mathbf{x_s}\}, \mathbf{y}, \mathbf{s}$	$\mathbf{X} = \{\mathbf{x}, \mathbf{x_s}\}, \mathbf{y}$	$\mathbf{X} = \{\mathbf{x}, \mathbf{x_s}\}$
▷Expected speed	Fast	Medium	Slow
▷Robustness	Unknown	High	Medium

an MDO formulation. A discussion of this notation leads nicely into the description of multi-level formulations, and more general hybrid formulations.

The terms used in this notation system are defined as follows:

SO System Optimizer

SA System Analyzer

$\mathbf{O}_i$ Subspace i optimizer

$\mathbf{A}_i$ Subspace i analyzer

$\mathbf{E}_i$ Subspace i evaluator

System optimizers and analyzers were introduced previously, but a subspace optimizer is a new concept. Subspace optimizers are the distinguishing feature of multi-level formulations, and are discussed in Section 3.4.2. Subspace analyzers determine the state variable values required satisfy governing equations for a subspace, and return response functions for a given design, while subspace evaluators merely evaluate the governing equations and return the pertinent residuals.

The type of execution is depicted by the following symbols:

[ ] Nested execution

|| Parallel execution

→ Sequential execution

This notation is helpful in describing the structure of an MDO formulation, however it is far from a complete description. Two formulations with identical notation under the above system may operate and perform very differently. Important distinguishing characteristics, such as how formulations handle system objective functions and subspace constraints, are not captured by this notation system.

A standardized naming scheme was also developed by Balling, expressing the formulation using a three-term name. The first term refers to whether the formulation has single or multiple optimization levels. The second term distinguishes whether SAND or NAND is used at the system level, and the third term makes the same distinction for the subspace level. Although the notation system introduced above applies to MDO formulations in general, the naming scheme is more limited. Examples given in the following sections will elucidate the notation system and naming scheme.

3.4.1 Single-Level Strategies

The three primary single-level formulations were described in detail previously, so only a brief illustration of how Balling's notation system and naming scheme applies to each will be given.

MDF

First consider the MDF formulation. It employs a single system level optimizer, a system analyzer, and subspace analyzers. In a two-subspace example the notation system would indicate MDF by:

$$SO[SA[A_1 \rightarrow A_2]]$$

The state variable determination is nested within each subspace analysis, and the subspace analyses are nested within the system analyzer. The appropriate name under Balling's convention is Single-NAND-NAND.

IDF

The IDF formulation again uses a single system level optimizer, and subspace analyzers. However, no system analyzer is used. A two-subspace IDF formulation in Balling's notation is depicted by:

$$SO[A_1||A_2]$$

The state variable determination is nested within each subspace analysis, and the subspace analyses are executed simultaneously by the system optimizer. The appropriate name is Single-SAND-NAND.

AAO

The AAO formulation utilizes a single system level optimizer, and subspace evaluators. A two-subspace AAO formulation in Balling's notation is depicted by:

$$SO[E_1||E_2]$$

The state variable determination is made by the system optimizer, and the subspace analyses are executed simultaneously by the system optimizer. The appropriate name is Single-SAND-SAND.

3.4.2 Multi-Level Strategies

In the single-level formulations above all design decisions, no matter how small, are made by a single system-level optimizer. This degree of design centralization may be overwhelming as the number of subspaces and interactions grow large. Distributing some of the decision-making authority throughout the system is an approach to alleviate problems associated with design centralization.

Some mature analysis tools are equipped with efficient optimization algorithms. It makes sense to utilize these integrated optimizers to make local decisions, rather than separating the analysis and forcing it to work with a system optimizer. Many design teams are led by a project manager (system level), but also by several group leaders or disciplinary experts. A single-level formulation bypasses the expertise of these individuals. Allowing experts or group leaders to make local decisions makes better use of existing resources. Local decision making is represented by subspace optimizers in multi-level architectures, providing the subspace autonomy that is frequently required to map a formulation to an existing organizational structure [4]. Multi-level formulations can reduce communication requirements, since local design variables no longer must be passed to the system optimizer. The system optimizer is limited to coordinating the subspace interactions and guiding the entire process to a system-optimal design.

Several variations of multi-level strategies have been developed. One class of these formulations is the *Disciplinary Constraint Feasible* (DCF) architecture (Figure 3.5). An optimizer is associated with each subspace, and is charged with ensuring any local design constraints are satisfied, and minimizing any applicable objective functions (which may be modified from the original design problem) with respect to subspace level variables. The system optimizer is responsible to ensure system consistency,

and minimize the system objective function, with respect to system level variables. The term DCF arises from the fact that each subspace optimizer is required to return a local design that is a consistent system that satisfies local 'discipline' design constraints. What variables are considered system and subspace level depends on the specific formulation. Details of how data is communicated in a DCF formulation is deferred until later sections. Collaborative Optimization is one example of a DCF architecture [10, 9], and is presented in detail in Chapters 5 and 6. Other examples of multilevel strategies in use are *Concurrent Subspace Optimization* (CSSO) [37, 42] and *Bi-Level Integrated System Synthesis* (BLISS) [39, 40].

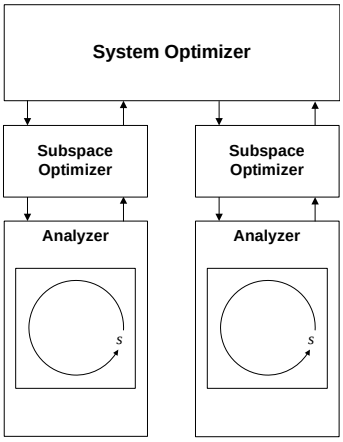

Figure 3.5: DCF structure.

Using Balling's naming scheme, a DCF formulation is termed a Multi-SAND-NAND approach. Using his notation system, the architecture shown in Figure 3.5 can be represented by:

$$SO[O_1[A_1]||O_2[A_2]]$$

It may be desirable to extend a multi-level formulation beyond two levels. Figure 3.6 illustrates one such structure. Methodologies such as *Analytical Target Cascading*

(ATC) exploit the hierarchical structure some systems possess[5]. A system optimizer fills the same roll as in the DCF architectures, but the intermediate subspace optimizers now are charged with coordinating any subspaces that are directly below them in the system hierarchy in addition to their own analysis. Please note that ATC is not an MDO formulation— it was developed as a product development tool rather than a design tool stemming from Multidisciplinary Analysis. ATC is described in detail in Chapters 5 and 7. This section serves as only an introduction to multi-level structures.

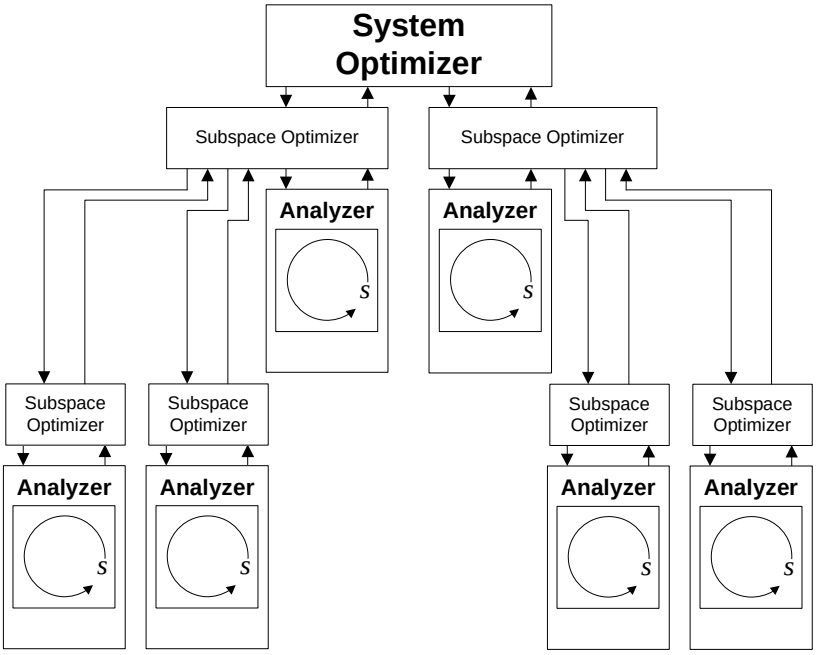

Figure 3.6: Multilevel optimization structure.

The multilevel optimization structure depicted in Figure 3.6 may be represented in Balling's notation as:

$$SO[O_1[A_1||O_3[A_3]||O_4[A_4]]||O_2[A_2||O_5[A_5]||O_6[A_6]]]$$

[5]Figure 3.6 does not depict ATC *per se*, but only a general hierarchical optimization structure.

3.4.3 Hybrid Strategies

Often a problem structure does not fit one of the standard formulations well. For example, one subspace may have its own optimizer, while the other subspaces do not. It may be tempting to force an existing problem structure to fit a particular MDO formulation because of familiarity with the formulation or for other reasons. This approach certainly can work, but may add several new variables and auxiliary constraints in the process. For some problems this may work fine with little penalty from the increased problem dimension. However, a grossly inefficient strategy may also result. In addition, breaking existing communication channels and forcing new channels in an organizational context may seriously disrupt a design team.

A viable alternative is to develop a hybrid formulation that is a more natural fit to the original design problem, preserving more of the existing structure and design tools. Although hybrid formulations are not discussed in detail in this thesis, they fall in line with the recurring theme of selecting an approach that best fits the design problem. Figure 3.7 illustrates one possible hybrid structure, whose representation in Balling's notation is:

$$SO[O_1[A_1]||A_2]$$

This chapter presented three fundamental Multidisciplinary Design Optimization Formulations— MDF, IDF, and AAO, and discussed the nature of each. All exhibit centralized decision making to some degree, and distributed analysis or evaluation. Multi-level formulations were introduced as a method for providing distributed decision making, improving how well MDO formulations map to existing design organizations. A general framework for describing MDO formulations was introduced. The

following chapter provides an in-depth comparison of two single-level formulations (MDF and IDF) by way of an engineering design problem.

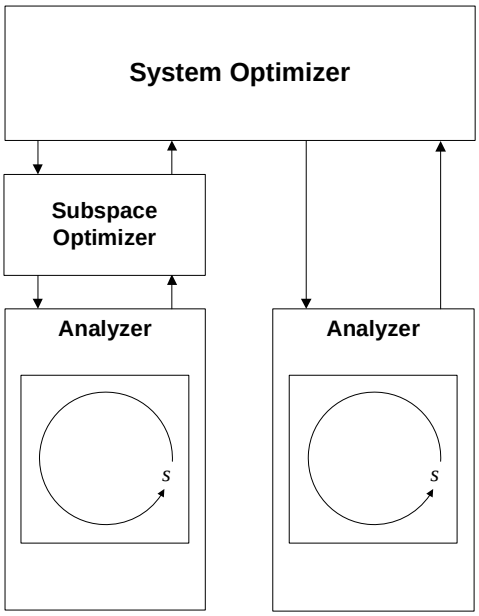

Figure 3.7: An example of a hybrid MDO architecture.

CHAPTER 4

Single Level Comparison: MDF vs. IDF

This chapter explores in detail two Multidisciplinary Design Optimizations presented in the previous chapter (MDF and IDF) by way of an engineering design example problem. MDF is the most basic formulation, requiring a complete analysis solution for every optimization iteration. IDF enables design and analysis to be performed simultaneously. Both MDF and IDF formulations are fully developed for the example problem, and how each formulation responds to increased coupling strength is explored.

4.1 Test Problem: Turbine Blade Design

The test problem chosen to perform a comparison between MDF and IDF is the turbine blade design problem introduced in Chapter 1. In this section the analysis model is fully developed for use in the comparison.

4.1.1 Test Problem Requirements

Before the turbine blade design problem was selected and developed, a set of requirements was identified. A suitable MDO test problem should possess the following characteristics:

- Inter-disciplinary coupling.

- Controllable coupling strength.
- Fully analytic model.
- Design tradeoffs.
- Physical significance.
- Manageable complexity.

Both MDF and IDF are strategies for solving design problems possessing interdisciplinary coupling, so this coupling is a required test problem characteristic for the comparison. A test problem with adjustable coupling strength facilitates the exploration of MDF and IDF performance variation with respect to system coupling strength.

Many MDO problems in practice involve computationally intensive computer simulations, such as finite element analysis (FEA), or computational fluid dynamics (CFD). MDO problems with such simulations pose special challenges, including significant computation time, response noise, and lack of derivative information. A fully analytic test problem enables the exploration of MDO formulations without these challenges. In addition, offering a detailed analytic model enables the reader to more easily replicate the results (typically not possible with complex system simulations).

The existence of design tradeoffs is also an important test problem property. A frequently encountered tradeoff is performance and cost, i.e. one component may be improved at the expense of degrading the other. Optimization problems without such tradeoffs are often trivial and sometimes unbounded. A well-designed example problem should possess some type of design tradeoff.

A purely abstract test problem would easily meet the above requirements, yet would lack physical significance. A connection to a real system aids in developing intuition for how the MDO formulations work, as well as demonstrating the practical

value of MDO. An easy to understand physical system with straightforward coupling behavior will meet our purposes well.

Scores of decomposable problems meet the above requirements, but may fail to be easily understood by an individual outside of a particular industry or discipline. An easily understood problem with manageable complexity will facilitate demonstration of MDO formulations to newcomers of this engineering discipline. Another source of problem complexity is the number of system interactions. A a problem is decomposed into n subspaces has $n(n-1)$ possible couplings. Restricting the number of disciplines to two or three will enable the reader to more easily follow and track the problem development and solution. Dozens of couplings would obfuscate the intuition of an introductory problem by burdening the reader with excessive details.

4.1.2 Turbine Blade Design Problem Description

Several physical systems were considered, and the design of a turbine blade used in a gas-turbine engine best met the requirements listed above. Simplifying assumptions were made to arrive at a fully analytic model, yet design performance trends and interactions were adequately captured in the model. A simplified diagram of a gas turbine engine is shown in Figure 4.1, and a drawing of a single turbine blade is shown in Figure 4.2.

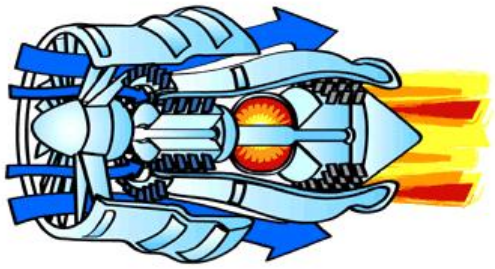

Figure 4.1: Diagram of a gas-turbine engine.

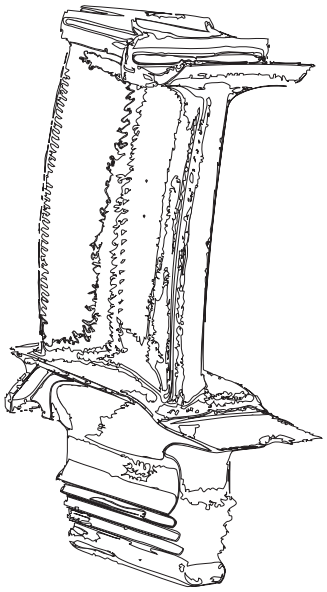

Figure 4.2: Diagram of a single gas-turbine blade.

A turbine blade in a gas turbine engine is exposed to high temperature combustion gasses moving at very high velocity. The blade is subject to high forces generated by aerodynamic drag force and centripetal acceleration. The blade must maintain structural integrity under these extreme conditions. In practice, modern alloys, such as Inconel, and advanced cooling systems are employed to ensure the durability of turbine blades. When a complete gas turbine system is modeled for design purposes, one possible objective is to maximize the thermal efficiency. Our simplified model cannot accurately predict this efficiency, but it can predict the mass m of the blade, and the heat transfer q through the blade. Intuitively both of these metrics should be minimized in order to maximize thermal efficiency. Without a complete model it is difficult to say how to balance the minimization of heat transfer and mass in order to best improve efficiency. This may be addressed with techniques for multi-objective optimization, discussed in Section 2.1.

Several phenomena were modeled in order to provide sufficient fidelity and to

capture the design tradeoffs and coupling behavior:

- Thermal expansion of the turbine blade in the axial direction.
- Stress and elongation due to centripetal acceleration.
- Aerodynamic drag force and the resulting bending stresses.
- Dependence of thermal conductivity (k), elastic modulus (E), and rupture stress (σ_r) on temperature.

It will be shown in the development of the mathematical model that the temperature profile of the blade is dependent upon the dilated length of the blade. Intuitively, if the blade elongates due to thermal expansion or centripetal forces, a larger surface area is exposed to the hot combustion gasses, affecting the heat transfer through the blade and the associated temperature profile. The model includes the dependence of the material's modulus of elasticity and thermal conductivity on temperature. Higher temperatures (caused by more exposed surface area) result in lower stiffness, causing greater elongation. In summary, temperature depends on length, and length depends on temperature (Figure 4.3).

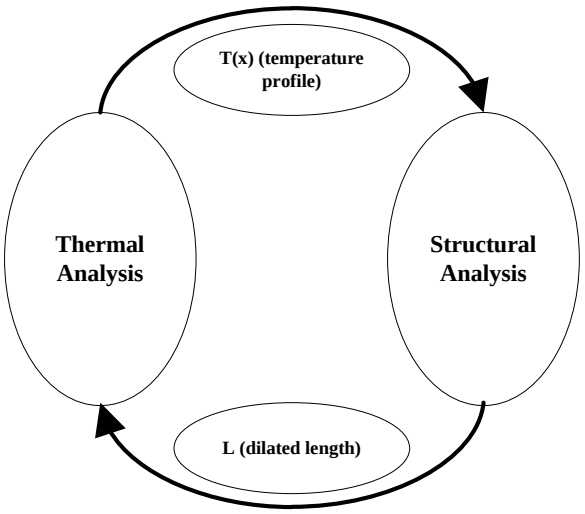

Figure 4.3: Analysis coupling present in the turbine blade design problem.

4.1.3 Mathematical Development

The mathematical model is decomposed by aspect into two subspaces— thermal and structural. The turbine blade is modeled as a simple rectangular fin in both disciplines, and is depicted in Figure 4.4. The design variables are the blade width w and thickness t. The blade has an initial undeformed length of L_0, and is subjected to combustion gas temperature T_g and velocity v_g. The blade is affixed to a rotor with angular velocity ω, resulting in a centripetal acceleration f_{ac}. x is the axial position measured from the attachment point of the blade to the rotor.

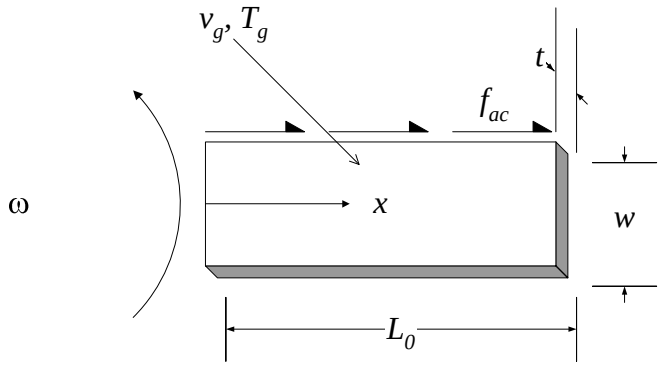

Figure 4.4: Turbine blade model schematic.

Four failure modes are considered. First is melting of the Inconel alloy used in the turbine blade. Second is interference between the blade and the turbine housing resulting from excessive elongation. The final two failure modes are structural failure from either the bending stress σ_b or axial stress σ_a exceeding the rupture stress σ_r at any point on the blade.

Several simplifying assumptions were made. The coefficient of thermal expansion α remains constant with temperature in this model, and it is assumed that no internal blade cooling is supplied. The inertial force f_{ac} is taken to be constant over the blade, since the blade length is much smaller than the rotor radius. Lateral contraction due

to the Poisson effect is neglected. Finally, the dependence of thermal conductivity (k), elastic modulus (E), and rupture stress (σ_r) on temperature is modeled with curve fits based on empirical data.

The complete turbine blade optimization problem is formulated as a multi-objective optimization problem shown below (equation 4.1):

$$\begin{aligned} \min_{\mathbf{X}=\{\mathbf{w},\mathbf{t}\}} \quad & \{q, m\} \\ \text{subject to} \quad & g_1(\mathbf{X}) = T_{max} - T_{melt} \leq 0 \\ & g_2(\mathbf{X}) = \delta_{total} - \delta_{allow} \leq 0 \\ & g_3(\mathbf{X}, x) = \sigma_a(x) - \sigma_r(T(x)) \leq 0 \\ & g_4(\mathbf{X}, x) = \sigma_b(x) - \sigma_r(T(x)) \leq 0 \\ & \{x | 0 \leq x \leq L_0 + \delta_{total}\} \end{aligned} \tag{4.1}$$

where T_{max} is the maximum temperature in the blade, T_{melt} is the melting temperature (solidus) of the blade alloy, δ_{total} is the blade elongation due to all effects modeled, δ_{allow} is the clearance between the blade and housing when cold, and $\sigma_a(x)$, $\sigma_b(x)$, and $\sigma_r(T(x))$ are the axial, bending, and rupture stresses along the blade. In practice the bending stress is far lower than the axial stress, so the simplification of considering the axial and bending stresses individually, rather than the combined out of plane stress, is validated. The mathematical development of each of the two disciplines follows.

Structural Analysis

The calculation of each of the functions used in equation 4.1 pertinent to structural analysis is outlined here. The structural objective function is the mass m of the turbine blade, determined with equation 4.2.

$$m = wtL_0\rho \tag{4.2}$$

The density of the turbine blade material is ρ. The next calculation is the total elongation δ_{total}. This requires the computation of both the thermal expansion δ_{th}, and the elongation due to axial acceleration δ_{ax}. The thermal expansion is dependent upon the change in temperature from the initial temperature T_0, and is derived as follows:

$$\begin{aligned} d\delta_{th} &= \alpha(T(x) - T_0)dx \\ \delta_{th} &= \int_0^{L_0} T(x)dx - \int_0^{L_0} \alpha T_0 dx \\ \delta_{th} &= \int_0^{L_0} T(x)dx - \alpha T_0 L_0 \end{aligned}$$

The last step is made because the coefficient of thermal expansion α is assumed constant with temperature. The temperature profile must be known in order to evaluate δ_{th}. At the beginning of a system analysis $T(x)$ is unknown because it is the output of the thermal analysis, and an initial guess must be made in order to begin the iterative Fixed Point Iteration analysis process (explained in §2.2), or a value supplied by the system optimizer if IDF is used.

The elongation due to centripetal acceleration is dependent on the angular velocity of the rotor ω and the radius of the rotor r. First the axial load as a function of axial position is calculated. The portion of the blade outboard of a position x pulls with load $P_a(x)$. The tangential velocity of the blade $v = \omega r$ is assumed to be constant over the blade length, and is valid if $L_0 \ll r$.

$$\begin{aligned} P_a(x) &= \int_x^{L_0+\delta_{total}} \frac{v^2}{r}\rho A_c dx \\ &= \frac{v^2}{r}\rho wt(L_0 + \delta_{total} - x) \\ &= \omega^2 r \rho wt(L_0 + \delta_{total} - x) \end{aligned}$$

The axial deflection δ_{ax} is found using the axial load, and then summed with the

thermal expansion δ_{th} to find the total elongation δ_{total} (equation 4.3).

$$\begin{aligned} \delta_{ax} &= \int_0^{L_0+\delta_{total}} \frac{P_a(x)dx}{A_c E(T(x))} \\ &= \omega^2 r\rho \int_0^{L_0+\delta_{total}} \frac{(L_0+\delta_{total}-x)}{E(T(x))}dx \end{aligned}$$

$$\begin{aligned} \delta_{total} &= \int_0^{L_0} T(x)dx - \alpha T_0 L_0 \\ &\quad +\omega^2 r\rho \int_0^{L_0+\delta_{total}} \frac{(L_0+\delta_{total}-x)}{E(T(x))}dx \end{aligned} \tag{4.3}$$

The integrations above require the value of the blade length after elongation $L = L_0 + \delta_{total}$, which is unknown before the integrations are performed. This is another coupling with the turbine blade length determination.

Now that the total deflection is known, the next responses to consider are the axial and bending stresses. The axial stress is a function of axial position, and is calculated with the relation $\sigma_a = P_a/A_c$, where P is the axial load and $A_c = wt$ is the cross sectional area as before.

$$\sigma_a(x) = \omega^2 r\rho(L-x) \tag{4.4}$$

The calculation of the bending stress is less trivial. First the aerodynamic drag force must be found.

$$P_{aero} = \frac{1}{2} A_f C_D \rho v^2$$

A_f is the frontal area, or wL, C_D is the drag coefficient, ρ is the combustion gas density, and v is the combustion gas velocity (assumed to be perpendicular to the blade). The velocity v is assumed to be constant, as is the combustion gas temperature and density. Therefore the only variables in the calculation of P_{aero} are

w and L. For convenience the constant $K = \frac{1}{2}C_D\rho v^2$ is defined, and the aerodynamic drag force is expressed as:

$$P_{aero} = KwL$$

Considering a position x on the blade, the total aerodynamic drag force acting on the blade outboard of that position is:

$$P_{aero}(x) = Kw(L - x)$$

and the bending moment at point x is:

$$M(x) = Kw\frac{(L-x)^2}{2}$$

Finally, using the beam stress formula $\sigma_b = Mc/I$ and the moment of inertia $I = wt^3/12$ for a rectangular beam, we arrive at the bending stress present in the blade at position x (equation 4.5).

$$\sigma_b(x) = \frac{3K(L-x)^2}{4t^2} \tag{4.5}$$

Thermal Analysis

A simple, one-dimensional heat transfer model was used to determine the temperature profile and heat transfer through the turbine blade. The model was derived beginning with the steady-state heat equation:

$$\frac{d^2T}{dx^2} + \left(\frac{1}{A_c}\frac{dA_c}{dx}\right) - \left(\frac{1}{A_c}\frac{h}{k}\frac{dA_c}{dx}\right)(T - T_\infty) = 0$$

The two boundary conditions used were a constant temperature base (where the blade attaches to the rotor), and an adiabatic tip condition. The combustion gas

temperature T_∞ was assumed constant. The dependence of thermal conductivity k on temperature is modeled with a curve fit (explained later in this section). It was assumed that k was constant over the entire blade length and depends only on the average blade temperature. The average convection coefficient $\overline{h}$ was approximated using empirical correlations involving the average Nusselt number $\overline{N_u}$ and the Prandtl number Pr.

$$\overline{N_u} = \frac{\overline{h}w}{k_g} = CRe_D^m Pr^{1/3}$$

The conduction coefficient k_g refers to the combustion gas, $Re_D = vw/\nu$ is the appropriate Reynold's number, and m is an empirical exponent of 0.731. C is the heat capacity of the combustion gas. Solving for $\overline{h}$, and substituting in appropriate values for the other parameters with SI units ($@T_\infty = 900°C$), the following expression results:

$$\overline{h}(v, w) = 9.196 v^{.731} w^{-.269}$$

The average convection coefficient $\overline{h}$ is function of only the blade width w and the combustion gas velocity v. Solving the heat equation with the appropriate boundary conditions, the following expressions for the temperature profile and the heat transfer through the blade into the rotor at the point of attachment result:

$$T(x) = \frac{\cosh(m(L-x))}{\cosh(mL)}(T_b - T_\infty) + T_\infty \tag{4.6}$$

$$q = wt(T_b - T_\infty)\tanh(mL)\sqrt{2\overline{h}(w+t)wtk} \tag{4.7}$$

where:

$$m = \sqrt{2h(t+w)/ktw}$$

Curve Fits

Several curve fits based on empirical data were employed in order to capture temperature dependence. All three parameters approximated have been explained previously, and the curve fits are summarized here.

The rupture stress σ_r was approximated using an *s-curve* function (equation 4.8). The rupture stress maintains a high value over most of the temperature range, but drops off steeply as the solidus temperature for Inconel X-750 is approached. Figure 4.5 illustrates this behavior.

$$\sigma_r(T) = \frac{1300}{1 + e^{.011(T-675)}} \tag{4.8}$$

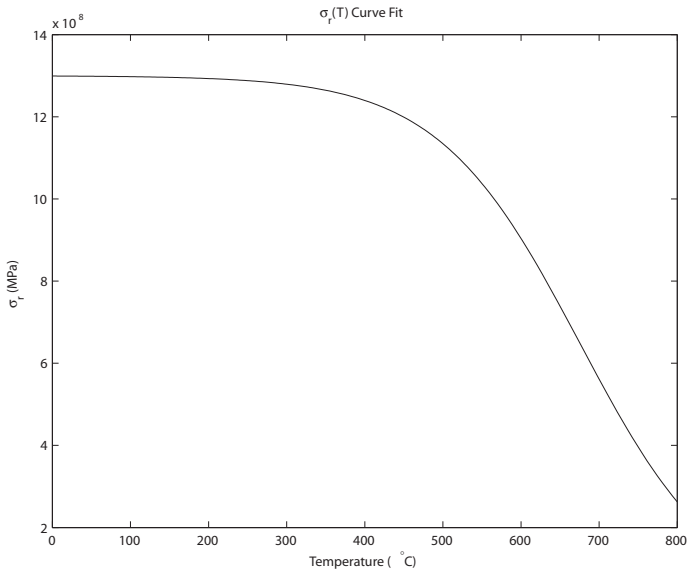

Figure 4.5: Curve fit for the rupture stress temperature dependence.

The conductivity of the blade k was modeled using a linear curve fit (equation 4.9). The dependence on average temperature $\overline{T}$ was captured from empirical data.

$$k(\overline{T}) = 6.8024 + 0.0172\overline{T} \tag{4.9}$$

The final curve fit was a fourth-order polynomial fit to the modulus of elasticity for the blade material (equation 4.10). Figure 4.6 shows the significant dependence that the modulus has on temperature. As shown in equation 4.3, this temperature dependence is utilized over the length of the blade to determine deflection.

$$E(T) = 209.8 - 0.0487T - .0002T^2 + 6 \cdot 10^{-7}T^3 - 6 \cdot 10^{-10}T^4 \tag{4.10}$$

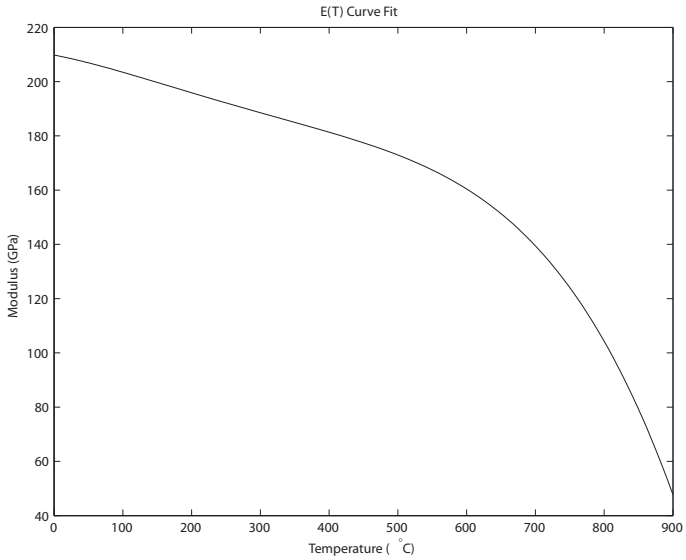

Figure 4.6: Curve fit for the elastic modulus temperature dependence.

4.1.4 Analysis Summary

The turbine blade analysis model is now fully developed, and is summarized in this section. Figure 4.7 illustrates the analysis problem structure, posed as a coupled, two-subspace system. The system has two shared design variables, and no local design variables, thus the entire design vector is $\mathbf{X} = \{w, t\}^T$. Please note the

difference between the design vector $\mathbf{X}$ and the position x, which is used to indicate that some of the responses are functions of position.

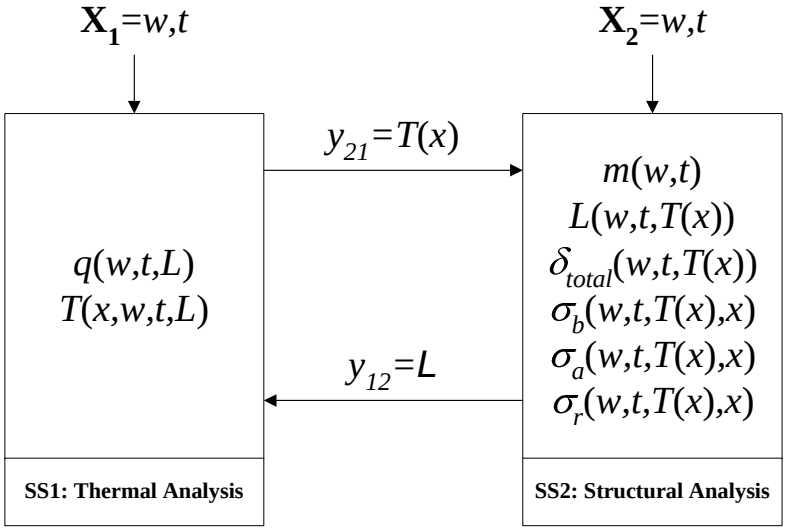

Figure 4.7: Diagram summarizing the turbine blade analysis.

The responses of the thermal analysis ($SS1$) are the heat loss q and the temperature distribution $T(x)$, both of which depend on the design vector (X) and the input from the structural analysis (coupling variable y_{21}), the dilated length L. The responses of the structural analysis ($SS2$) are the mass m, dilated length L, total deflection δ_{total}, and the bending, axial, and rupture stresses σ_b, σ_a, and σ_r. These responses in general depend upon the design vector (X) and the input from the thermal analysis (coupling variable y_{12}), the temperature distribution $T(x)$.

Figure 4.7 is similar to Figure 4.3 in that they both illustrate the nature of the interdisciplinary coupling (Figure 4.7 provides more detail). To employ Fixed Point Iteration (section 2.2) for analysis of a turbine blade design, the design vector (X) is held fixed, and an initial guess is made for one of the coupling variables (such as the temperature distribution). The sequential subspace evaluation is iterated until the change in coupling variable value is less than some small value ε at each iteration.

Table 4.1 shows the analysis results for a particular design. Function valued responses such as temperature distribution and stress distribution are displayed in Figure 4.8.

Table 4.1: Turbine blade analysis results.

parameters	values	variables	values	responses	values
ρ	8510 kg/m^3	w	0.08 m	q	0.2046 W
L_0	0.05 m	t	0.005 m	m	0.1702 kg
α	$12.6 \cdot 10^{-6}$ m/K			L	0.507 m
r_b	0.5 m			δ	0.007 m
ω	$2100 rad/s$				
δ_{max}	0.05 m				
ρ_g	3.522 kg/m^3				
C_d	2.0				
v	100 m/s				
T_b	300 $°C$				
T_g	900 $°C$				
ε	$1.0 \cdot 10^{-8}$				

Note that the bending stress is insignificant compared to the axial stress, and at no point along the blade does any stress value exceed the rupture stress. The temperature is also feasible (and will never be infeasible since the gas temperature is lower than the alloy's melting temperature).

Now that the analysis has been completed, it can be used for the next step—optimization.

4.2 Turbine Blade Design with MDF

The turbine blade design was optimized first using the MDF approach (§3.1). The FPI analysis procedure described in the previous section was used to find an analysis

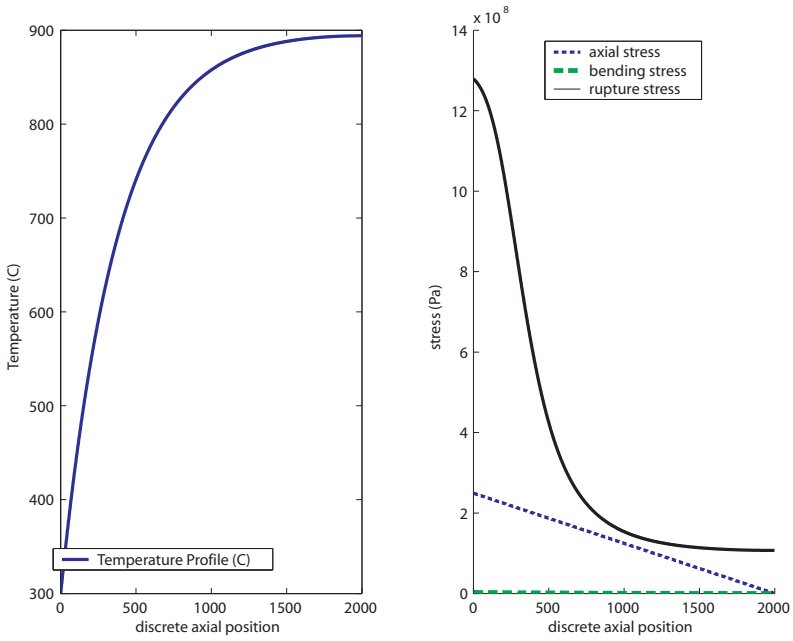

Figure 4.8: Temperature and stress response of the turbine blade analysis.

solution (consistent set of coupling variables) at every optimization iteration during the MDF process. The optimizer controlled only the design variables, $\mathbf{X} = \{w, t\}^T$. Since the design problem has two objective functions (m and q), one (m) was treated as a constraint function, and the other (q) was treated as the objective function. The MDF formulation follows.

$$\begin{aligned} \min_{\mathbf{X}=\{\mathbf{w},\mathbf{t}\}} \quad & q \\ \text{subject to} \quad & g_1(\mathbf{X}) = T_{max} - T_{melt} \leq 0 \\ & g_2(\mathbf{X}) = \delta_{total} - \delta_{allow} \leq 0 \\ & g_3(\mathbf{X}, x) = \sigma_a(x) - \sigma_r(T(x)) \leq 0 \\ & g_4(\mathbf{X}, x) = \sigma_b(x) - \sigma_r(T(x)) \leq 0 \\ & g_5(\mathbf{X}, x) = m - m_{max} \leq 0 \\ & \{x | 0 \leq x \leq L_0 + \delta_{total}\} \end{aligned}$$

The mass was constrained to not exceed 0.04 kg. The parameter values from Table 4.1 were used, and the optimal design was found to be $\{w_*,\ t_*\}^T = \{0.0131,\ 0.0075\}^T$.

4.3 Turbine Blade Design with IDF

The turbine blade design was then optimized using the slightly more sophisticated approach, IDF (§3.2). The primary difference in this demonstration is the optimizer not only controls the design variables, $\mathbf{X} = \{w, t\}^T$, but also provides input values for the coupling variables, L and $T(x)$. In addition, two equality constraints were added to the optimization to ensure that the values supplied for the coupling variables matched the outputs from the subspaces that calculated the coupling variables. A result is intermediate designs are not usually consistent systems. Only at convergence will all constraints be met. The IDF formulation is shown below.

$$
\begin{aligned}
\min_{\mathbf{X}=\{\mathbf{w},\mathbf{t}\},T(x),L} \quad & q \\
\text{subject to} \quad & g_1(\mathbf{X}) = T_{max} - T_{melt} \leq 0 \\
& g_2(\mathbf{X}) = \delta_{total} - \delta_{allow} \leq 0 \\
& g_3(\mathbf{X}, x) = \sigma_a(x) - \sigma_r(T(x)) \leq 0 \\
& g_4(\mathbf{X}, x) = \sigma_b(x) - \sigma_r(T(x)) \leq 0 \\
& g_5(\mathbf{X}, x) = m - m_{max} \leq 0 \\
& g_6(\mathbf{X}, x) = T(x) - T(\mathbf{X}, x) = 0 \\
& g_7(\mathbf{X}, x) = L - L(\mathbf{X}) = 0 \\
& \{x | 0 \leq x \leq L_0 + \delta_{total}\}
\end{aligned}
$$

The parameter values from Table 4.1 were again used. The optimal design found using IDF was basically identical to the MDF results: $\{w_*,\ t_*\}^T = \{0.0128,\ 0.0074\}^T$.

When applied to this simple example, both MDF and IDF are successful in finding the optimum system design. One may wonder why go through the additional complexity of IDF. Important reasons for choosing IDF such as opportunity for parallelization were discussed previously, and the next section explores another reason—the effect of coupling strength on solution performance.

4.4 Effects of Coupling Strength

The effect that E and α have on coupling strength was already discussed qualitatively. Here a quantitative approach is used to illustrate the effect of coupling strength on both the MDF and IDF approaches. This is an excellent means to compare the suitability of these two approaches for various design problems based on the system coupling strength.

Twenty different turbine blade design optimizations were performed for both MDF and IDF. Each design had a different multiplier for the modulus of elasticity E. Smaller multipliers resulted in a more *flexible* blade, and a more strongly coupled system. The computation time was recorded for each optimization. The same investigation could have been performed with multipliers greater than one for α. The results of this experiment are displayed in Figure 4.9.

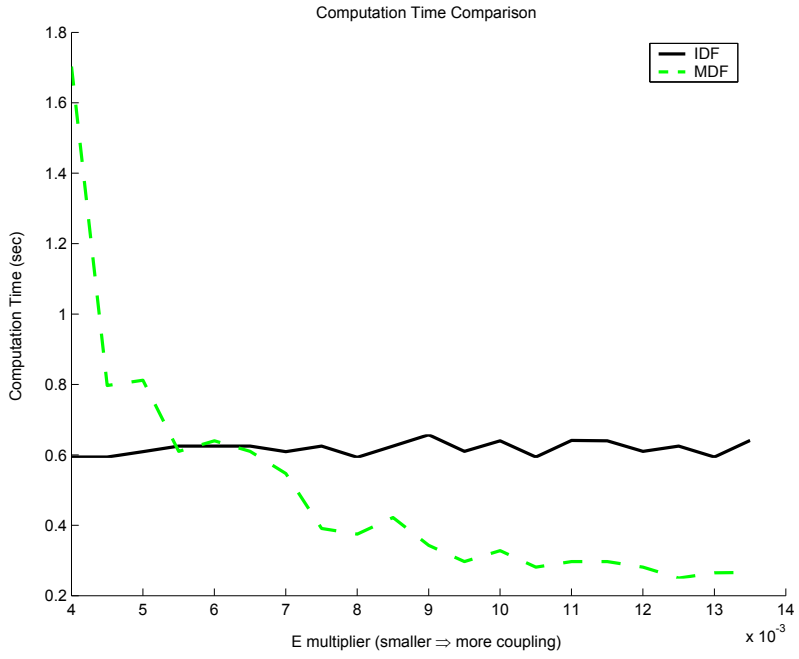

Figure 4.9: Comparison of MDF and IDF solution time as a function of coupling strength.

The plot elucidates the robustness of IDF with respect to coupling strength, and the sensitivity of MDF to the same factor. Weakly coupled systems are more efficiently solved with MDF, while strongly coupled systems require excessive iterations for the inner analysis loops for MDF. The computation time required for the IDF approach is virtually constant for all levels of coupling strength investigated here. Also, the computation time for IDF did not reflect any parallel processing. If parallel processing had been used the IDF computation time would be a little more than 50% of the values shown in Figure 4.9.

The turbine blade design problem explored throughout this chapter was an effective means to illustrate the details of both the MDF and IDF formulations. The test problem possessed intuitive feedback coupling, and had the means to easily modify the system's coupling strength. IDF was shown to be a more efficient choice for strongly coupled problems. However, IDF does have shortcomings. In an organizational context such a centralized approach is less than desirable. Typically design groups are lead by group leaders or disciplinary experts, and frequently mature areas of design have developed optimization procedures. IDF cannot utilize these resources since all decisions are made at the top level. Multilevel methods are required to do so. The remainder of the thesis introduces and compares a few selected multilevel methods.

CHAPTER 5

Formulation of Selected Multilevel Strategies

The previous two chapters introduced several motivating factors for the development of multilevel optimization strategies (i.e., strategies that employ subspace optimizers). Some of the more important factors include distributing design authority throughout an organization and utilizing existing design and optimization tools. This chapter presents the formulation of two selected multilevel strategies: Collaborative Optimization (CO), and Analytic Target Cascading (ATC). CO is an MDO approach, inspired by Multidisciplinary Analysis needs, and is frequently used for aerospace applications. ATC is a product development tool intended for use at the beginning of a design process, and has ties to the automotive industry.

An emphasis of this thesis is to clarify the distinctions between the CO process and the ATC process. The formulations are presented with such a comparison in mind. Although CO and ATC were developed based on very different needs, their mathematical formulations are remarkably similar. Through some manipulation CO can be made to appear very similar to ATC, and visa-versa. However, the solution process is very different for each, and the requirements each methodology is based upon colors their respective natures. For example, CO is a Multidisciplinary Design Optimization (MDO) method, inspired by the work done in Multidisciplinary Anal-

ysis (MDA). The result is that CO formulation implicitly emphasizes analysis. In addition, because systems that CO is intended for are typically partitioned by aspect (discipline), the bi-level architecture of CO is considered adequate since disciplines are all considered to be members of the same level. In contrast, ATC is not an MDO method, but rather originated from product development needs in industries that are organizationally structured in a hierarchical manner with more than two levels. A formulation that allows multiple levels is therefore required.

A source of confusion regarding CO and ATC is the terminology for each methodology. Effort is given here to solidify the meaning of the terminology for each strategy and to clarify the differences. For example, both strategies use 'targets', yet the targets are composed of different quantities and are treated differently in CO and ATC. Also, CO (as with other MDO formulations) categorize some quantities as coupling variables, and ATC has a category called linking variables. These terms sound similar, but are demonstrated in this chapter to refer to different things.

Chapter 6 and 7 demonstrate the CO and ATC design processes, respectively, using an example problem that can be formulated as a coupled system (for CO demonstration) or as a hierarchical system (for ATC demonstration).

Both CO and ATC have been the subject of continued research, and several refinements for each strategy have been presented in the literature [1, 12, 14, 16]. As this thesis is an initial comparison, only the basic formulation of each strategy will be presented, facilitating a clear comparison of the fundamental processes. Comparisons involving more sophisticated formulations is left for future work.

5.1 Collaborative Optimization

Collaborative Optimization was developed in response to industry needs not met by then current MDO formulations. For example, strategies such as IDF and AAO (Chapter 3) are highly centralized, requiring that a single system optimizer (or project leader), make all design decisions no matter how small. This section develops the basic formulation of Collaborative Optimization, defined as a *bi-level hierarchic disciplinary constraint feasible approach*[1]. Much of the material here was originally presented in [9] and [10].

5.1.1 Collaborative Optimization Overview

To improve the practical value and industry utilization of MDO strategies, several issues needed to be addressed, including:

- Subspace autonomy.
- High initial investments in restructuring design organizations.
- Flexibility to changes made during the design process.

Existing design organizations commonly have specialized design resources, such as specialized optimization procedures (such as found in control, structures, and aerodynamic disciplines) or experienced design experts. Centralized strategies such as IDF or AAO bypass these resources and pass all decision making to a central system optimizer. This not only stifles design group productivity, but often results in heavy data communication requirements. Centralized strategies do not map well to complex organizations with distributed decision making authority. The analysis integration required to implement a centralized strategy can be a substantial challenge, and may need to be repeated if the design process changes.

[1] See §3.4.2 for an introduction to disciplinary constraint feasible (DCF) formulations.

Collaborative optimization addresses these issues by employing subspace optimizers and a unique treatment of system level and subspace level design problems. Subspace optimizers allow the inclusion of legacy design tools and disciplinary experts, promote subspace autonomy (local design freedom), and reduce system communication requirements. Subspaces aim at meeting targets for shared and coupling variables set by the system optimizer, while maintaining subspace design feasibility. The system optimizer performs inter-subspace and system-subspace negotiation via these targets, i.e., the system optimizer manages these targets to guide the subspaces into agreement and toward the system level objective.

System level targets are the only inbound communication a subspace receives. Subspaces do not need to communicate with each other. A subspace optimizer's function is to meet its targets and satisfy its constraints. For example, if a higher fidelity subspace analysis is brought onboard partway through the design, CO can cope with this change since all the subspace has to do is meet its targets. In this way CO can cope with the progressively increasing complexity throughout the design process.

As with most MDO formulations, CO may be applied to general non-hierarchic, arbitrarily coupled systems. Auxiliary constraints that require matching of system targets decouple the subspaces, and allow general systems to fit the hierarchic structure of CO. This property provides industry with a new class of solution architectures that may provide a more natural fit to many existing hierarchic organizational structures. However, if an organization has more than two levels in its hierarchy (such as with many organizations partitioned by object), it must be restructured to match the CO architecture. Other strategies that allow for more levels may be preferable in this case.

5.1.2 Collaborative Optimization Formulation

To introduce the CO formulation, a schematic of a system with N subspaces is presented in Figure 5.1. Several details are omitted in the schematic for clarity of the structure, but will be introduced shortly. The figure illustrates the hierarchic communication channels between the system optimizer and the subspaces.

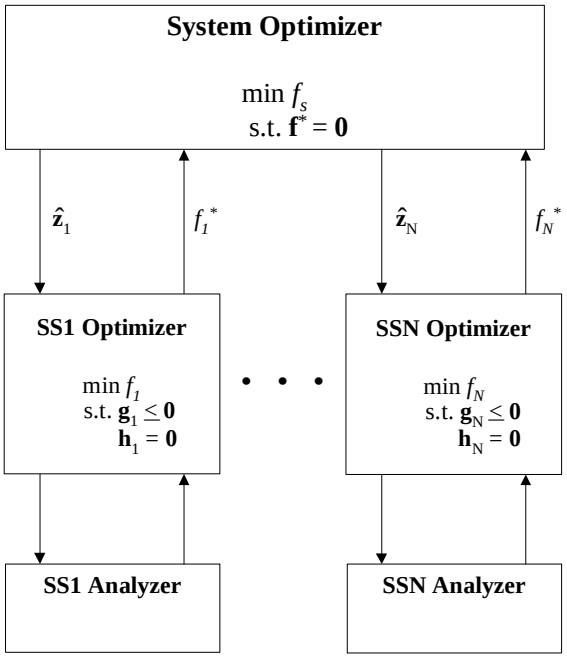

Figure 5.1: Simplified Collaborative Optimization architecture.

The system optimizer seeks to minimize the system objective function f_s, subject to the auxiliary constraints that require the optimal values of the subspace objective functions ($\mathbf{f}^* = f_1^*, f_1^* \ldots f_N^*$) to all be zero. The system optimization is performed with respect to the system targets $\hat{\mathbf{z}}$. The optimizer for subspace i receives the system targets pertinent to subspace i, and seeks to minimize the discrepancy between the targets $\hat{\mathbf{z}}_i$ and the corresponding quantities generated by the i^{th} subspace $\mathbf{z}_i$, subject to the subspace's local design constraints $\mathbf{g}_i$ and $\mathbf{h}_i$. The discrepancy between the system targets and their corresponding subspace quantities is captured by the i^{th}

subspace objective function, f_i.

With the general CO architecture known, the relevant terminology can be formally introduced. As first described in section 2.1.4, the design variables of the original design problem $\mathbf{X}$ are separated into local and shared variables according to the system partitioning.

$$\mathbf{X} = \{\mathbf{x}_i^T, \mathbf{x}_s^T\}^T \quad i = 1 \ldots N$$

The original design variable vector may be separated into quantities that are design variables for a particular subspace, called subspace design variables ($mathbf{X}_i$). Subspace design variables $mathbf{X}_i$ contain design variables that are local to subspace i ($\mathbf{x}_i$), and shared variables that are required inputs to subspace i ($\mathbf{x}_{si}$). Subspace design variables do not include any coupling variables, since subspace design variables are members of the original design vector, and coupling variables are artifacts of decomposition and are never part of the original design vector.

$$\mathbf{X}_i = \{\mathbf{x}_i^T, \mathbf{x}_{si}^T\}^T \quad \forall\, i$$

The system objective function $f_s(\hat{\mathbf{z}})$ is a scalar function of the system targets. It might be computed directly by one of the subspaces, or could be a function of some combination of subspace responses and design variables.

The system targets $\hat{\mathbf{z}}$ are the decision variables of the system-level optimization, and are parameters in the subspace optimizations. These targets manage the subspace interactions, that is they guide all shared and coupling variables. Each subspace is passed a set of locally pertinent targets $\hat{\mathbf{z}}_i$, that are a part of the complete system target vector $\hat{\mathbf{z}}$. The $\hat{\mathbf{z}}_i$ in general do have common components, although the special case exists where the local targets do not have common components. The collection of all local targets comprises the complete system target vector $\hat{\mathbf{z}}$.

A target in the vector $\hat{\mathbf{z}}_i$ exists for every shared variable $\mathbf{x}_{si}$ used in subspace i and for every input coupling variable $\mathbf{y}_{ij}$ and output coupling variable $\mathbf{y}_{ji}$[2]. The values in $\mathbf{z}_i$ are those generated by subspace i, each of which corresponds to a target in $\hat{\mathbf{z}}_i$. Note that the local design variables $\mathbf{x}_i$ are not members of $\mathbf{z}_i$. This contributes to subspace autonomy, and a reduced number of system level auxiliary constraints (compared to AAO or IDF).

$$\mathbf{z}_i = \{\mathbf{x}_{si}^T, \mathbf{y}_{ij}^T, \mathbf{y}_{ji}^T\}^T$$

The subspace objective functions f_i quantify the deviation of the $\mathbf{z}_i$ values calculated by subspace i from the corresponding targets $\hat{\mathbf{z}}_i$, typically through the square of the Euclidean norm. The subspace objective functions f_i are functions of the subspace design variables and the subspace coupling variables. The vector of local targets $\hat{\mathbf{z}}_i$ is considered a fixed parameter in subspace optimizations. The optimal value of the subspace objective function f_i^* is passed to the system level problem and used in the system-level auxiliary equality constraints.

$$f_i(\mathbf{X}_i, \mathbf{y}_{ij}, \mathbf{y}_{ji}) = ||\mathbf{z}_i - \hat{\mathbf{z}}_i||_2^2$$

Design constraints from the original design problem are partitioned into subspace constraints ($\mathbf{g}_i(\mathbf{X}_i, \mathbf{y}_{ij})$, $\mathbf{h}_i(\mathbf{X}_i, \mathbf{y}_{ij})$) that are functions only of subspace design variables and subspace input coupling variables, and into system level constraints that require design variables from multiple subspaces and perhaps subspace responses as inputs ($\mathbf{g}_s(\hat{\mathbf{z}})$, $\mathbf{h}_s(\hat{\mathbf{z}})$). The latter are treated as functions of system targets, since they are to be met solely by the selection of system target values. Other inputs to these functions, such as subspace responses, are functions of targets in the CO formulation from the system perspective.

[2]The definition of shared and coupling variables is provided in section 2.1.4.

Now that the terminology has been established, the detailed formulation of the system and subspace problems can be presented.

System Level Formulation

$$\begin{aligned} \min_{\hat{\mathbf{z}}} \quad & f_s(\hat{\mathbf{z}}) \\ \text{subject to} \quad & \mathbf{f}^*(\hat{\mathbf{z}}) = \mathbf{0} \\ & \mathbf{g}_s(\hat{\mathbf{z}}) \leq \mathbf{0} \\ & \mathbf{h}_s(\hat{\mathbf{z}}) = \mathbf{0} \end{aligned} \tag{5.1}$$

Subspace Formulation

$$\begin{aligned} \min_{\mathbf{X}_i, \mathbf{y}_{ij}} \quad & f_i(\mathbf{X}_i, \mathbf{y}_{ij}) = ||\mathbf{z}_i - \hat{\mathbf{z}}_i||_2^2 \\ \text{subject to} \quad & \mathbf{g}_i(\mathbf{X}_i, \mathbf{y}_{ij}) \leq \mathbf{0} \\ & \mathbf{h}_i(\mathbf{X}_i, \mathbf{y}_{ij}) = \mathbf{0} \end{aligned} \tag{5.2}$$

At convergence the system is consistent, and has attained system optimality. Subspaces require no communication from other subspaces, only system targets passed down from the system optimizer. Note that in some cases subspaces communicate quantities to the system level other than f_i^*, such as when required to compute system level functions f_s, $\mathbf{g}_s$, or $\mathbf{h}_s$. These system level functions are still only functions of the system targets when viewed from the system perspective, since the system supplies only these targets and receives all necessary information to calculate these functions.

Note that as with other multi-level MDO formulations, the optimization is nested. That is, at every iteration of the system-level optimizer, each subspace optimizer must perform a complete optimization. This can lead to a large number of function evaluations, but since CO is a bi-level formulation the nesting is normally manageable.

5.2 Analytical Target Cascading

Many design problems and organizations possess a purely hierarchical problem structure with feedforward communication. Design problems with this structure may be solved with a multi-level strategy called Analytical Target Cascading (ATC). ATC is the result of efforts to formalize activities early in the product development process, particularly for the automotive industry. This section introduces the ATC process and its mathematical formulation. The dissertation by Kim [23] formally presented the ATC methodology, and [24] provides an excellent summary of ATC.

5.2.1 Analytical Target Cascading Overview

Management of a company producing complex products may set top-level performance targets for a product based on marketing or other criteria. Attempting to generate a product design that meets these targets while abiding by engineering constraints may be difficult or impossible if the design problem is viewed as a single monolithic task. Some design problems allow partitioning into a purely hierarchic structure, as illustrated in Figure 5.2. The nomenclature used in this figure will be explained shortly. With this type of partitioning, appropriate targets may be set for each of the smaller design tasks such that if met, the system as a whole will be a consistent, feasible design that meets the top-level system targets.

Once appropriate targets are known for each subspace, the individual design tasks can be executed in parallel and autonomously. Communication requirements between subspaces are removed, speeding up the design process. The interaction between subspaces is considered when the targets are set. Once the targets are set, the design groups can proceed with detailed analysis and design.

Analytical Target Cascading is the process of determining the correct targets for

each subspace for a specified set of system-level targets. Simulations used in the ATC process must be detailed enough to capture salient interactions, yet computationally inexpensive to allow ATC to be performed quickly enough to be useful as an early product development tool. If such models do not exist, surrogate models of high-fidelity simulations can be generated, or response surfaces made from experimental data.

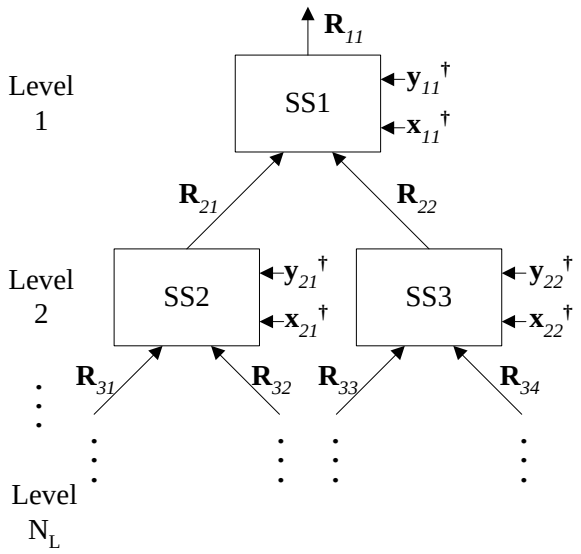

Figure 5.2: General hierarchic partitioning structure.

Implementation of ATC in a product development process is summarized below.

1. Specify top-level targets.
2. Propagate targets through the system.
   (a) Develop low-fidelity analysis models.
   (b) Partition the system.
   (c) Formulate the target cascading problem.
   (d) Solve the target cascading problem with a coordination strategy.
3. Perform subspace detail design to meet targets.

4. Verify system consistency and system level target matching.

If subspace targets cannot be met then the process is repeated beginning at step 2 with updated information obtained from the previous design attempt. If the top-level targets cannot be met, management is presented with the design that best matches the targets, and is charged with the decision to accept the design or update the targets and repeat the entire process.

Before the ATC process is explained, the nomenclature and structure presented in Figure 5.2 is defined. A parent subspace takes as inputs its local decision variables, linking variables, and child problem responses. It then returns its own set of responses. In a hierarchical analysis structure child problems cannot require inputs from parent problems, and links that skip levels are not permitted. This structure provides only for feedforward coupling, since feedback would require a parent to send a response to a child. The original design variables and coupling variables are categorized differently in the ATC process. It is important that these new categories are defined carefully to prevent continued confusion between ATC and MDO strategies. Recall the following definitions of MDO quantities:

$\mathbf{x}$ *Local design variables*: Design variables that are each inputs to only one subspace.

$\mathbf{x}_s$ *Shared design variables*: Design variables that are each inputs to more than one subspace.

$\mathbf{y}$ *Coupling variables*: Quantities that are passed from one subspace to another that are not original design variables, but rather artifacts of decomposition.

With these in mind the terms specific to ATC are defined. The † symbol is used to distinguish the ATC terms when a symbol already used for MDO nomenclature is employed.

$\mathbf{y}^\dagger$ *Linking variables*: Quantities that are input to more than one subspace. These could be either shared variables (original design variables) or coupling variables (not original design variables).

$\mathbf{x}^\dagger$ *Local decision variables*: Variables that a particular subspace determines the value of. May or may not be original design variables.

$\mathbf{R}$ *Responses*: Values generated by subspaces required as inputs to respective parent subspaces. May or may not be coupling variables.

$\mathbf{T}$ *Targets*: Values set by parent subspaces to be matched by the corresponding quantities from child subspaces. Targets may exist for either responses or linking variables.

The first subscript of the terms shown in Figure 5.2 (e.g. $\mathbf{R}_{ij}$) indicates the level i of the pertinent subspace. The set E_i contains all subspaces at the i^th level, and the second subscript j indicates what element of E_i the pertinent subspace is (the within-level element numbers). Every subspace j within level i has a set of child problems, defined as C_{ij}.

The mapping between MDO and ATC nomenclature is clarified by the Venn diagram in Figure 5.3. What in the MDO context was called coupling variables may be either linking variables or responses in an ATC context. Shared variables from MDO are always linking variables in an ATC formulation. These distinctions will be made more clear in the presentation of the ATC formulation and in the illustrative example of Chapter 7.

ATC targets ($\mathbf{T}$ correspond to slightly different quantities than do CO targets. In CO, targets ($\mathbf{z}$) are restricted to shared and coupling variables. ATC targets may be set for responses that are neither shared nor coupling variables. ATC targets are used

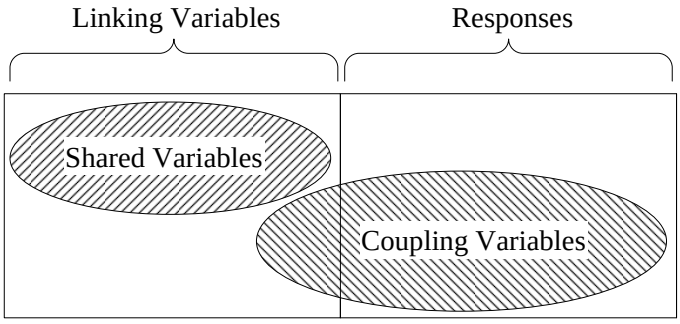

Figure 5.3: Venn diagram of ATC/CO nomenclature mapping.

to ensure system consistency (similar to CO targets) by ensuring shared and coupling variables match. Targets for lower level problems are also set in order to ensure that the higher level problem is capable of meeting its own targets. Linking variables or responses that must match are coordinated by a common parent subspace.

One possible coordination strategy between subspaces consists of nested loops. The top level problem is solved first, sending targets its corresponding child problems. After the child problems are solved, the required responses are sent to the parent problem, which is then solved again. Note that each child problem may have in turn associated child problems, requiring inner coordination loops. The process is repeated until the top level problem converges. This strategy has demonstrated convergence properties [23, 30], however may require many optimization executions. This motivates the requirement for low-fidelity analysis models. This is a key process difference between CO and ATC. ATC has a nested coordination strategy, while CO has nested optimizations. A nested coordination strategy means that while parent problems are usually executed multiple times, these problems are provided a static set of responses from child problems and a static set of targets from higher level problems. Each optimization problem can be executed autonomously, without waiting for results of lower optimization problems at every optimization iterations

(as is the case with CO). In summary, CO requires a single execution of the top level problem while ATC requires many, and CO has nested optimization while ATC has nested coordination.

5.2.2 Analytical Target Cascading Formulation

The mathematical formulation of top level, intermediate, and bottom level problems are delineated here. To simplify notation, the subscripts *sup*, *s*, and *ss* indicate the supersystem, system, and subsystem levels. Each problem is identified by a problem label $P_{i,j}$, where i and j indicate the level and within-level element numbers respectively[3]. The intermediate level problem is the most general since it both receives targets from above and responses from below.

Top Level Problem
$\mathbf{P}_{sup}$

$$\min_{\mathbf{x}^{\dagger}{}_{sup}=\{\mathbf{x}_{sup},\mathbf{y}^{\dagger}{}_{s},\mathbf{R}_s,\varepsilon_R,\varepsilon_y\}} \quad ||\mathbf{R}_{sup}-\mathbf{T}_{sup}|| + \varepsilon_R + \varepsilon_y \tag{5.3}$$

$$\text{subject to} \quad \sum_{k\in C_{sup}} ||\mathbf{R}_{s,k}-\mathbf{R}^L_{s,k}|| \le \varepsilon_R$$

$$\sum_{k\in C_{sup}} ||\mathbf{y}^{\dagger}{}_{s,k}-\mathbf{y}^{\dagger L}_{s,k}|| \le \varepsilon_y$$

$$g_{sup}(\mathbf{x}_{sup},\ \mathbf{R}_s) \le 0$$

$$h_{sup}(\mathbf{x}_{sup},\ \mathbf{R}_s) \le 0$$

Intermediate Level Problem
$\mathbf{P}_{s,j}$

$$\min_{\mathbf{x}^{\dagger}{}_{s,j}=\{\mathbf{x}_{s,j},\mathbf{y}^{\dagger}{}_{s,j},\mathbf{y}^{\dagger}{}_{ss},\mathbf{R}_{ss},\varepsilon_R,\varepsilon_y\}} \quad ||\mathbf{R}_{s,j}-\mathbf{R}^U_{s,j}|| + ||\mathbf{y}_{s,j}-\mathbf{y}^U_{s,j}|| + \varepsilon_R + \varepsilon_y \tag{5.4}$$

$$\text{subject to} \quad \sum_{k\in C_{s,j}} ||\mathbf{R}_{ss,k}-\mathbf{R}^L_{ss,k}|| \le \varepsilon_R$$

$$\sum_{k\in C_{s,j}} ||\mathbf{y}^{\dagger}{}_{ss,k}-\mathbf{y}^{\dagger L}_{ss,k}|| \le \varepsilon_y$$

$$g_{s,j}(\mathbf{x}_{s,j},\ \mathbf{y}^{\dagger}{}_{s,j},\ \mathbf{R}_{s,j}) \le 0$$

$$h_{s,j}(\mathbf{x}_{s,j},\ \mathbf{y}^{\dagger}{}_{s,j},\ \mathbf{R}_{s,j}) \le 0$$

[3]The second subscript is not necessary for the top level problem since only one element exists at the top level.

Bottom Level Problem
$\mathbf{P}_{ss,j}$

$$\begin{aligned} \min_{\mathbf{x}^{\dagger}{}_{ss,j}=\{\mathbf{x}_{ss,j},\mathbf{y}^{\dagger}{}_{ss,j}\}} \quad & ||\mathbf{R}_{ss,j} - \mathbf{R}^{U}_{ss,j}|| + ||\mathbf{y}_{ss,j} - \mathbf{y}^{U}_{ss,j}|| \\ \text{subject to} \quad & g_{s,j}(\mathbf{x}_{s,j},\ \mathbf{y}^{\dagger}{}_{s,j},\ \mathbf{R}_{s,j}) \leq 0 \\ & h_{s,j}(\mathbf{x}_{s,j},\ \mathbf{y}^{\dagger}{}_{s,j},\ \mathbf{R}_{s,j}) \leq 0 \end{aligned} \tag{5.5}$$

The top level problem controls its local design variables, shared variables from lower problems, desired responses from its child problems, and the allowable compatibility deviations (ε_R and ε_y) with the objective of minimizing the difference between its responses and the top level targets, and minimizing the allowable compatibility deviations. Local design constraints must be satisfied, as well as the response and linking variable compatibility constraints. It is these compatibility constraints that guide the lower level problem into agreement with respect to responses matching targets and linking variables being equal. The desired response values and linking variable values from child problems ($\mathbf{R}_{s,k}$, $\mathbf{y}^{\dagger}{}_{s,k}$) are decision variables for the top level problem. The values of $\mathbf{R}_{s,k}$ and $\mathbf{y}^{\dagger}{}_{s,k}$ that solve the top level problem are passed to child problems as targets, and are treated as parameters in the child problems.

The intermediate level problem is similar, but also aims at matching its linking variables to the linking variable targets set by its parent problem (which is not necessarily the top level problem). The linking variables pertinent to this intermediate problem $\mathbf{y}^{\dagger}{}_{s,j}$, not just those for child problems, are included in the set of decision variables $\mathbf{x}^{\dagger}{}_{s,j}$. The bottom level problem has no child problems, and therefore has no compatibility constraints to meet. It only must meet local design constraints while matching targets from its parent problem. Because of this, bottom level problems have the most design freedom. Many possible solutions can exist that both match targets exactly while satisfying local design constraints. At higher levels in the hierarchy design freedom is progressively reduced, until it is a minimum at the top level.

Staying feasible and meeting targets can be a struggle for the top level problem.

As noted earlier, the top level optimization is performed multiple times (as opposed to Collaborative Optimization's single top level execution). What is used as stopping criteria? One successful strategy is to iterate the top problem until the sum of the allowable compatibility deviations (tolerances) ε_R and ε_y converge asymptotically, and the relative change of this sum falls below a prescribed value. This recognizes that ATC may not be capable of exactly meeting the system level targets since the top level target deviations are not included in this criterion, and provides a way for the process to terminate if no compatible design exists that will meet the system level targets. If the tolerances are not sufficiently close to zero at convergence, new targets must be set and the process repeated since the solution was not a compatible design. In CO the compatibility constraints must be satisfied with strict equalities, rather than within some tolerance. This can lead to convergence issues in some problems solved with CO.

CHAPTER 6

Collaborative Optimization Example

The Collaborative Optimization architecture for complex system optimization was presented in Chapter 5. This chapter introduces a new example design problem, and then uses Collaborative Optimization to solve the design problem.

6.1 Statically Indeterminate Structural Analysis Test Problem

In order to demonstrate the Collaborative Optimization and Analytical Target Cascading methodologies in detail, a second test problem[1] was developed— a statically indeterminate structural analysis (SISA) of a beam and rod system. The SISA problem shares several attributes with the turbine blade design problem, such as being a fully analytic model, having physical significance, and possessing tradeoffs. However, it differs in that it is expandable to an arbitrarily large number of subspaces, and can be manipulated algebraically into a tightly coupled formulation suitable for utilization of MDO strategies (including CO), or into a purely hierarchical formulation suitable for ATC. With this test problem both CO and ATC can be used to solve the same design problem (in a manner natural for both CO and ATC), enabling a clear illustration of the similarities and differences between the two strategies.

[1]The first test problem in this thesis (design of a turbine blade) was developed in Section 4.1.

This section develops the analytical model for the SISA test problem, and then illustrates how it can be posed as a coupled system. Fixed Point Iteration (Section 2.2) is employed to demonstrate system analysis, and the MDF optimization strategy (Section 3.1) is used to solve the design problem to establish a baseline approach. The SISA problem is posed as a hierarchical problem in Section 7.1.1 for use in illustrating Analytical Target Cascading.

6.1.1 SISA Problem Description

Figure 6.1 exhibits the physical setup of the SISA problem. A number of cantilevered beams (n_b) are attached to a vertical wall. Each of the beams have a solid circular cross section of unique diameter d_i, where i is the beam number. All beams are of a uniform length L. The rods interconnecting the beams also have solid circular cross sections of diameter d_{rj}, where j is the rod number. All rods are of unique length l_{rj}, and are attached to the beams with pin joints. There are $n_r = n_b - 1$ rods in the system. A downward force F_1 is applied at the pin joint of beam 1.

A physical application of such a configuration is an anchoring system. Imagine several steel posts set in a concrete foundation. Multiple posts are used in order to distribute the load more evenly over the foundation, preventing damage to the concrete.

The system is statically indeterminate because the number of unknown forces and moments is greater than the number of available equilibrium equations. Some type of compatibility constraints must be employed to enable solution of the system.

Several assumptions are made for the analysis of this system, including:

- The system remains linear.
- Body forces (gravity) are neglected.
- Shear stress in the beams is neglected.

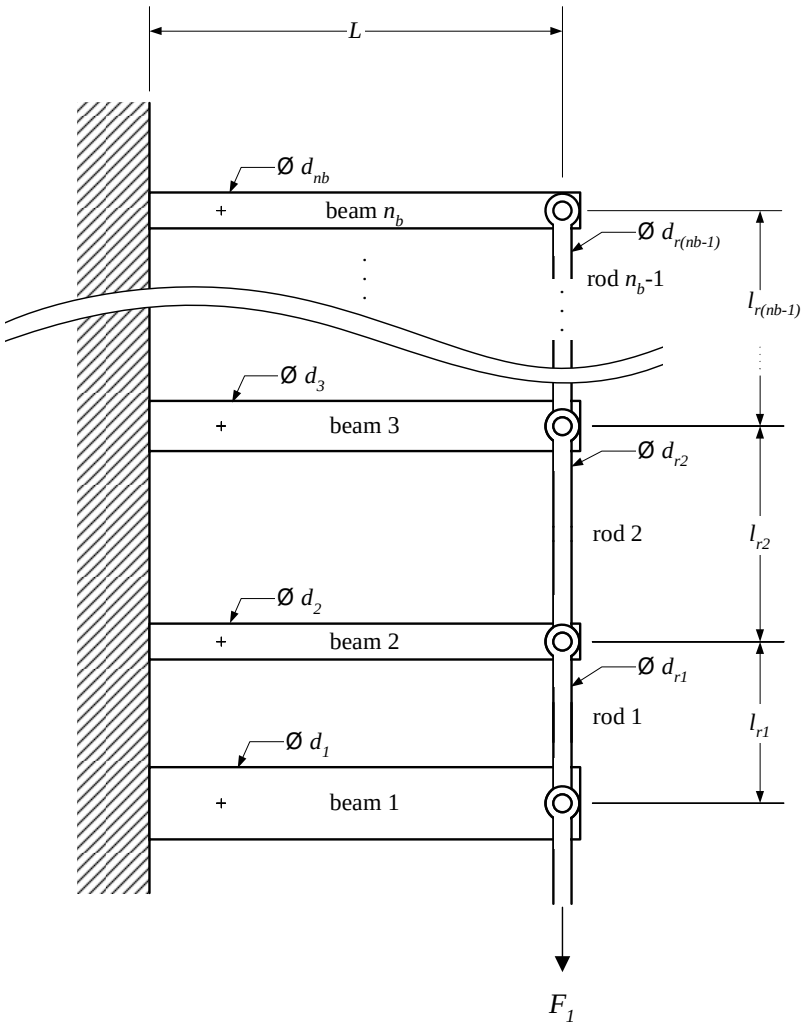

Figure 6.1: Schematic of the statically indeterminate structural analysis test problem.

- Buckling is not considered (rods in tension only).
- Stress concentrations are neglected.
- Homogeneous material properties.

The design problem is posed as a mass allocation problem. The system mass must remain below a prescribed limit, and the objective is to distribute the mass such that the deflection of the end of beam 1 under a given force F_1 is minimized subject to stress constraints. As is, an optimization will allocates all available mass to a single beam. Which beam receives the mass depends on the starting design.

If the initial design specifies equal beam diameters, beam 1 receives all the mass, and the other beam diameters will be driven to zero (or lower bounds if used). This is the most efficient design, and the global optimum. If the initial design specifies one of the beams to be larger than the others, that beam will receive all available mass. Intuitively, if a beam is already large, adding mass to it is more effective than distributing it elsewhere since rigidity increases in a quartic manner with diameter (quadratically with mass). A system of n beams has n local minima; which minimum is found depends on the starting point.

A monotonic design problem such as this does not make for a very interesting example. To resolve this issue consider the physical application. A reason for using more than one rod in an anchoring system is to distribute the load over more of the foundation to prevent failure of the foundation material. Addition of a *maximum transmitted force* constraint[2] models this requirement.

Table 6.1 summarizes what model quantities are considered design variables and design parameters. The next section describes one possible partitioning of the SISA design problem, and the separation of local and shared variables is made with respect to this partitioning.

Table 6.1: SISA test problem variables and parameters.

local design variables	x_i:	d_i	$i = 1 \ldots n_b$
shared design variables	x_{si}:	d_{rj}	$j = 1 \ldots n_r$
design parameters	$\mathbf{p}$:	$L, l_{rj}, E, \rho, \sigma_{allow}, m_{allow}, Ft_{allow}$	$j = 1 \ldots n_r$

Ft_{allow} is the allowable transmitted force, ρ is the material density, σ_{allow} is the allowable stress, and E is the modulus of elasticity. The design optimization problem

[2]This is mathematically equivalent to a maximum transmitted moment constraint since $M = FL$ and L is constant.

in negative-null form is presented in equation 6.1. Recall that the collection of all design variables X is the union of the shared variables $\mathbf{x_s}$ and the local variables $\mathbf{x}$.

$$\begin{aligned} \min_{\mathbf{X}=\{\mathbf{x},\mathbf{x_s}\}} \quad & f(\mathbf{X}) = \delta_1(\mathbf{X}) \\ \text{subject to} \quad & g_{1i}(\mathbf{X}) = \sigma_{bi}(\mathbf{X}) - \sigma_{allow} \leq 0 \quad i = 1 \ldots n_b \\ & g_{2j}(\mathbf{X}) = \sigma_{aj}(\mathbf{X}) - \sigma_{allow} \leq 0 \quad j = 1 \ldots n_r \\ & g_3(\mathbf{X}) = \sum_{i=1}^{n_b} m_{bi} + \sum_{i=1}^{n_r} m_{rj} - m_{allow} \leq 0 \\ & g_{4i}(\mathbf{X}) = Ft_i - Ft_{allow} \leq 0 \quad i = 1 \ldots n_b \end{aligned} \tag{6.1}$$

The responses of interest in the analysis include the deflection of each beam end (δ_i), the extension of each rod (δ_{rj}), the bending stress in each beam (σ_{bi}), the axial stress in each rod (σ_{aj}), the transmitted force at each beam (Ft_i), and the beam and rod masses (m_{bi} & m_{ri}). From fundamental solid mechanics theory the following relations were used in the development of this model:

$$\sigma_b = \frac{Mc}{I}$$
$$I = \frac{\pi}{64} d^4$$
$$\sigma_a = \frac{P}{A_c}$$
$$\delta_b = \frac{PL^3}{3EI}$$
$$\delta_r = \frac{PL}{EA_c}$$

M is the bending moment in a beam at the point of wall attachment, I is the area moment of inertia, P is the load, and A_c is the cross-sectional area. To be more precise with the load quantities, F_i is designated as the downward force exerted at the end of beam i, and F_j is the axial load present in rod j (which is equal to the force applied to the beam above rod j, F_{j+1}). The responses of interest may be generalized and grouped into three categories— *intermediate or bottom beam*, *arbitrary rod*, and *top beam*.

Intermediate or Bottom Beam

$$\delta_i = \frac{64L^3(F_i - F_{i+1})}{3\pi E d_i^4} \tag{6.2a}$$

$$\sigma_{bi} = \frac{32L(F_i - F_{i+1})}{\pi d_i^3} \tag{6.2b}$$

$$m_{bi} = \frac{\pi}{4} d_i^2 L\rho \tag{6.2c}$$

Arbitrary Rod

$$\delta_{rj} = \frac{4F_{j+1}l_{rj}}{\pi E d_{rj}^2} \tag{6.3a}$$

$$\sigma_{aj} = \frac{4F_{j+1}}{\pi d_{rj}^2} \tag{6.3b}$$

$$m_{rj} = \frac{\pi}{4} d_{rj}^2 l_{rj}\rho \tag{6.3c}$$

Top Beam

$$\delta_{n_b} = \frac{64L^3 F_{n_b}}{3\pi E d_{n_b}^4} \tag{6.4a}$$

$$\sigma_{bn_b} = \frac{32LF_{n_b}}{\pi d_{n_b}^3} \tag{6.4b}$$

$$m_{bn_b} = \frac{\pi}{4} d_{n_b}^2 L\rho \tag{6.4c}$$

Note that the transmitted force Ft_i for each beam is $F_i - F_{i+1}$ (F_i for the top beam). Compatibility conditions require that the deflection of the end of beam i (measured to a fixed frame) is equal to the deflection of the beam above $(i+1)$ plus the extension of the connecting rod $(j = i)$.

$$\delta_i = \delta_{i+1} + \delta_{ri} \tag{6.5}$$

6.1.2 MDO Specific Formulation

One possible solution strategy is to view the SISA problem as a coupled system. The resulting formulation is well suited for solution by MDO methods. The deflection of a beam is dependent upon the deflection of the surrounding beams, introducing coupling into the system. A beam and the rod connected to it from above is considered a subspace. The top beam is of course a subspace by itself. A three-beam case was chosen for developing the MDO specific formulation in order to ensure an intermediate beam was employed, while maintaining simplicity. The three-beam case may easily be extended to any number of beams.

Figure 6.2 illustrates the decomposition of the three beam case into three subspaces, and shows the functional relationships and subspace communication paths. This moderately coupled example has four of a possible $3(3-1) = 6$ possible subspace interactions. The rest of this section develops the details of the functional relationships found in this decomposition.

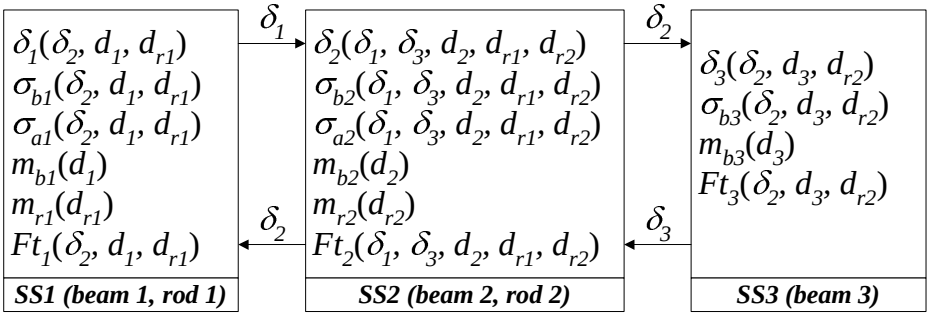

Figure 6.2: MDO decomposition of the three-beam SISA problem.

Using the MDO nomenclature developed earlier, the design and coupling variables are identified for each subspace in Table 6.2.

The total design vector is therefore:

Table 6.2: Design and coupling variables for each subspace in the SISA test problem.

$SS1$	$SS2$	$SS3$
$y_{21} = \delta_1$	$y_{21} = \delta_1$	$y_{32} = \delta_2$
$y_{12} = \delta_2$	$y_{23} = \delta_3$	$y_{23} = \delta_3$
$x_1 = d_1$	$y_{12} = \delta_2$	$x_31 = d_3$
$x_{s1} = d_{r1}$	$y_{32} = \delta_2$	$x_{s2} = d_{r2}$
	$x_2 = d_2$	
	$x_{s1} = d_{r1}$	
	$x_{s2} = d_{r2}$	

$$\mathbf{X} = \left\{ \begin{array}{c} \mathbf{x} \\ \\ \mathbf{x_s} \end{array} \right\}$$

where:

$$\mathbf{x} = \{d_1\ d_2\ d_3\}^T$$
$$\mathbf{x}_s = \{d_{r1}\ d_{r2}\}^T$$

The development of all functional relationships depicted in Figure 6.2 follows.

Subspace 1

Objective: given the inputs δ_2, d_1, and d_{r1}, and parameters $\mathbf{p}$, find the responses δ_1, σ_{b1}, σ_{a1}, m_{b1}, and m_{r1}. From equations 6.3a and 6.5 we know:

$$\delta_1 = \delta_2 + \delta_{r1} = \delta_2 + \frac{4F_2 l_{r1}}{\pi E d_{r1}^2}$$

Using equation 6.2a:

$$\delta_1 = \frac{64L^3(F_1 - F_2)}{3\pi E d_1^4} = \delta_2 + \frac{4F_2 l_{r1}}{\pi E d_{r1}^2}$$

If we define the two constants for convenience, $C_{1a} = \frac{64L^3}{3\pi E d_1^4}$ and $C_{1b} = \frac{4l_{r1}}{\pi E d_{r1}^2}$, the force F_2 may be solved for, and then the displacement δ_1:

$$F_2 = \frac{-\delta_2 + C_{1a}F_1}{C_{1b} + C1a} \tag{6.6a}$$

$$\delta_1 = C1a\,(F_1 - F_2) \tag{6.6b}$$

With a value for F_2, the other responses follow from equations 6.2b, 6.3b, 6.2c, and 6.3c respectively:

$$\sigma_{b1} = \frac{32L(F_1 - F_2)}{\pi d_1^3} \tag{6.6c}$$

$$\sigma_{a1} = \frac{4F_2}{\pi d_{r1}^2} \tag{6.6d}$$

$$m_{b1} = \frac{\pi}{4} d_1^2 L\rho \tag{6.6e}$$

$$m_{r1} = \frac{\pi}{4} d_{r1} l_{r1} \rho \tag{6.6f}$$

Subspace 2

Objective: given the inputs δ_1, δ_3, d_2, d_{r1}, and d_{r2}, and parameters $\mathbf{p}$, find the responses δ_2, σ_{b2}, σ_{a2}, m_{b2}, and m_{r2} . From equations 6.3a and 6.5 we know:

$$\delta_2 = \delta_1 - \delta_{r1} = \delta_1 - \frac{4F_2 l_{r1}}{\pi E d_{r1}^2}$$

Using equation 6.2a:

$$\delta_2 = \frac{64L^3(F_2 - F_3)}{3\pi E d_2^4} = \delta_1 - \frac{4F_2 l_{r1}}{\pi E d_{r1}^2}$$

If we define the three constants for convenience, $C_{2a} = \frac{64L^3}{3\pi E d_2^4}$, $C_{2b} = \frac{4l_{r1}}{\pi E d_{r1}^2}$, and $C_{2c} = \frac{4l_{r2}}{\pi E d_{r2}^2}$, and use compatibility again (equation 6.5), the following two equations for the displacement δ_2 result:

$$\delta_2 = C_{2a}(F_2 - F_3) = \delta_1 - F_2 C_{2b}$$

$$\delta_2 = C_{2a}(F_2 - F_3) = \delta_3 + F_3 C_{2c}$$

If these equations are solved simultaneously for F_2 and F_3, δ_2 may then be found:

$$F_2 = \frac{\delta_1 - \delta_3 - C_{2c}F_3}{C_{2b}} \tag{6.7a}$$

$$F_3 = \frac{\delta_1 - \delta_3 - \frac{\delta_3 C_{2b}}{C_{2a}}}{C_{2c} + C_{2b} + \frac{C_{2c}C_{2b}}{C_{2a}}} \tag{6.7b}$$

$$\delta_2 = C2a\left(\frac{\delta_1 - \delta_3 - C_{2c}F_3}{C_{2b}} - F_3\right) \tag{6.7c}$$

With values for F_2 and F_3, the other responses follow from equations 6.2b, 6.3b, 6.2c, and 6.3c respectively:

$$\sigma_{b2} = \frac{32L(F_2 - F_3)}{\pi d_2^3} \tag{6.7d}$$

$$\sigma_{a2} = \frac{4F_3}{\pi d_{r2}^2} \tag{6.7e}$$

$$m_{b2} = \frac{\pi}{4} d_2^2 L \rho \tag{6.7f}$$

$$m_{r2} = \frac{\pi}{4} d_{r2} l_{r2} \rho \tag{6.7g}$$

Subspace 3

Objective: given the inputs δ_2, d_3, and d_{r2}, and parameters $\mathbf{p}$, find the responses δ_3, σ_{b3}, and m_{b3}. From equations 6.3a and 6.5 we know:

$$\delta_3 = \delta_2 - \delta_{r2} = \delta_2 - \frac{4F_3 l_{r2}}{\pi E d_{r2}^2}$$

Using equation 6.4a:

$$\delta_3 = \frac{64L^3F_3}{3\pi E d_3^4} = \delta_2 - \frac{4F_3 l_{r2}}{\pi E d_{r2}^2}$$

If we define the two constants for convenience, $C_{3a} = \frac{64L^3}{3\pi E d_3^4}$ and $C_{2c} = \frac{4l_{r2}}{\pi E d_{r2}^2}$, the force F_3 may be solved for, and then the displacement δ_3:

$$F_3 = \frac{\delta_2}{C_{3a} + C_{3b}} \tag{6.8a}$$

$$\delta_3 = C_{3a}\left(\frac{\delta_2}{C_{3a} + C_{3b}}\right) \tag{6.8b}$$

With a value for F_2, the other responses follow from equations 6.2b, 6.3b, 6.2c, and 6.3c respectively:

$$\sigma_{b3} = \frac{32LF_3}{\pi d_3^3} \tag{6.8c}$$

$$m_{b3} = \frac{\pi}{4} d_3^2 L\rho \tag{6.8d}$$

6.1.3 Analysis and MDF Results

The analysis of the three-beam SISA problem was performed using Fixed Point Iteration (Section 2.2), and then this procedure was employed in a MDF formulation to solve the design problem stated in equation 6.1. The Fixed Point Iteration analysis is detailed below. The superscript k indicates the iteration number of a coupling variable y^k, and the difference between coupling variable values between iterations is $\Delta y = y^{k+1} - y^k$. In this problem the coupling variables are δ_1, δ_2, and δ_3. The norm $||\Delta\delta_i||$ is the Euclidean norm of the difference vector $\{\Delta\delta_1\ \Delta\delta_2\ \Delta\delta_3\}^T$.

Iteration 0 :

guess δ_2^0, δ_3^0.

Iterate until $||\Delta\delta_i|| < \varepsilon$

Iteration 1 :

$$\delta_1^1 = \delta_1(\delta_2^0)$$

$$\delta_2^1 = \delta_2(\delta_1^1, \delta_3^0)$$

$$\delta_3^1 = \delta_3(\delta_2^1)$$

Iteration 2 :

$$\delta_1^2 = \delta_1(\delta_2^1)$$

$$\delta_2^2 = \delta_2(\delta_1^2, \delta_3^1)$$

$$\delta_3^2 = \delta_3(\delta_2^2)$$

$$\vdots$$

As per the discussion in Section 2.2.1, the superscript i indicates the iteration number, and ε is the maximum inconsistency allowed between subspaces. A stopping criteria of $\varepsilon = 1 \cdot 10^{-6}$ was used[3]. The FPI algorithm required 94 iterations for convergence, indicating strong coupling behavior. With design variable and parameter values given in Table 6.3, the resulting deflection δ_1 was 5.60 millimeters.

Table 6.3: SISA design variable and parameter values for FPI analysis.

design variable	value	units	parameter	value	units
d_1	0.05	meters	F_1	1000	Newtons
d_2	0.05	meters	L	1.000	meters
d_3	0.05	meters	l_{r1}	1.000	meters
d_{r1}	0.005	meters	l_{r2}	1.000	meters
d_{r2}	0.005	meters	E	70	GPa
			ρ	2700	kg/meter^3

[3]Although this value for ε may appear small, it is reasonably sized given the magnitude of the δ_i values.

To begin the optimization, stress, mass and transmitted force limits were set as $\sigma_{allow} = 127\ MPa$, $m_{allow} = 7\ kg$, and $Ft_{allow} = 400\ N$. The MDF formulation of the problem is identical to the original design problem, given in equation 6.1. Using the parameter values and the starting design point from Table 6.3, the optimal design was found to be:

$$\mathbf{X}_* = \{d_1\ d_2\ d_3\ d_{r1}\ d_{r2}\}^T = \{.035\ .035\ .029\ .005\ .003\ \}^T$$

The minimum deflection at beam 1 was $\delta_1^* = 27.0$ millimeters. Beams 1 and 2 could not be made any larger without violating the transmitted force constraint. The mass constraint was also active, but none of the stress constraints were active. If the transmitted force limit is set to a value more than half of the applied force F_1, the resulting optimal design allocates zero mass to the third beam. In general, additional beams are required only if the existing beams cannot distribute the force well enough to prevent failure of the foundation. If the applied force F_1 is less than the transmitted force limit, then only one beam is required. The mass of any other beams in the system will be zero.

6.2 CO Formulation

Here the SISA problem is structured to fit the Collaborative Optimization architecture. Please refer to Section 5.1 for the details of the general CO formulation. The partitioning of the SISA problem presented in section 6.1.2 was preserved for the CO implementation, as were all parameter values utilized in the MDF solution of Section 6.1.3.

To communicate the general structure of the CO implementation, a simplified diagram of the structure is provided in Figure 6.3. The constraint indexing is consistent with equation 6.1.

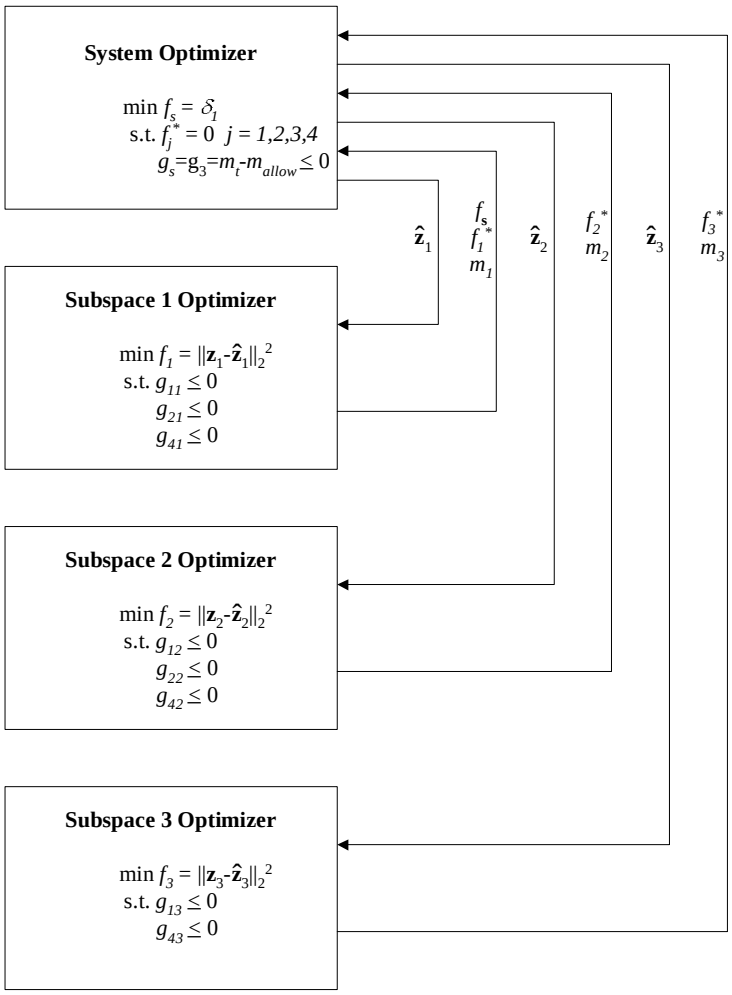

Figure 6.3: Simplified diagram of the three-beam SISA CO formulation.

Each subspace is provided the required system targets $\hat{\mathbf{z}}_i$ by the system optimizer. These targets are design variables at the system level, and parameters at the subspace level. Each subspace returns to the system optimizer the minimum possible deviation from the system targets (f_i^*), the mass of all components in the subspace (m_i), and in the case of subspace 1 the system objective function $f_s = \delta_1$. The subspace masses are required to evaluate the system-wide total mass constraint, $g_s = g_3$. Each subspace is responsible to satisfy its own local bending and axial stress constraints (g_{1i} and g_{2i} respectively), and the maximum transmitted force constraint (g_{4i}).

As explained previously, the system targets $\hat{\mathbf{z}}$ in the CO formulation consist of all shared and coupling variables. At convergence these targets must be matched by the corresponding quantities generated by the subspaces, $\mathbf{z}$. The *hat* designation indicates that a quantity is a target set by the system optimizer. The actual $\mathbf{x}_i$, $\mathbf{x}_{si}$, and $\mathbf{y}_{ij}$ vectors were defined for the SISA problem in Table 6.2. The system targets are defined as:

$$\hat{\mathbf{z}} = \{x_{s1},\ x_{s2},\ y_{12},\ y_{21},\ y_{23},\ y_{32}\}^T$$

$$\begin{aligned}
\hat{\mathbf{z}}_1 &= \{x_{s1},\ y_{12},\ y_{21}\}^T \\
\hat{\mathbf{z}}_2 &= \{x_{s1},\ x_{s2},\ y_{12},\ y_{21},\ y_{23}\ y_{32}\}^T \\
\hat{\mathbf{z}}_3 &= \{x_{s2},\ y_{23},\ y_{32}\}^T
\end{aligned}$$

The CO formulation of the SISA design problem is presented in equations 6.9–6.12. The system optimization is performed with respect to the system targets, and the subspace optimization is performed with respect to all inputs to the subspace analyses. In a normal sequential design process, the input coupling variables $\mathbf{y}_{ij}$ are fixed parameters. However, in the CO formulation they become decision variables.

System Optimizer

$$\begin{aligned}
\min_{\hat{\mathbf{z}}} \quad & f_s(\hat{\mathbf{z}}) \\
\text{subject to} \quad & \mathbf{f}^*(\hat{\mathbf{z}}) = \mathbf{0} \\
& \mathbf{g}_s(\hat{\mathbf{z}}) = \textstyle\sum_{i=1}^{3} m_i - m_{allow} \leq \mathbf{0}
\end{aligned} \tag{6.9}$$

Subspace 1 Optimizer

$$\begin{aligned}
\min_{\mathbf{X}_1, \mathbf{y}_{1j}} \quad & f_1(\mathbf{X}_1, \mathbf{y}_{1j}) = ||\mathbf{z}_1 - \hat{\mathbf{z}}_1||_2^2 \\
\text{subject to} \quad & \mathbf{g}_{11}(\mathbf{X}_1, \mathbf{y}_{1j}) \leq \mathbf{0} \\
& \mathbf{g}_{21}(\mathbf{X}_1, \mathbf{y}_{1j}) \leq \mathbf{0} \\
& \mathbf{g}_{41}(\mathbf{X}_1, \mathbf{y}_{1j}) \leq \mathbf{0}
\end{aligned} \tag{6.10}$$

Subspace 2 Optimizer

$$\begin{aligned} \min_{\mathbf{X}_2,\mathbf{y}_{2j}} \quad & f_2(\mathbf{X}_2,\mathbf{y}_{2j}) = ||\mathbf{z}_2 - \hat{\mathbf{z}}_2||_2^2 \\ \text{subject to} \quad & \mathbf{g}_{12}(\mathbf{X}_2,\mathbf{y}_{2j}) \leq \mathbf{0} \\ & \mathbf{g}_{22}(\mathbf{X}_2,\mathbf{y}_{2j}) \leq \mathbf{0} \\ & \mathbf{g}_{42}(\mathbf{X}_2,\mathbf{y}_{2j}) \leq \mathbf{0} \end{aligned} \tag{6.11}$$

Subspace 3 Optimizer

$$\begin{aligned} \min_{\mathbf{X}_3,\mathbf{y}_{3j}} \quad & f_3(\mathbf{X}_3,\mathbf{y}_{3j}) = ||\mathbf{z}_3 - \hat{\mathbf{z}}_3||_2^2 \\ \text{subject to} \quad & \mathbf{g}_{13}(\mathbf{X}_3,\mathbf{y}_{3j}) \leq \mathbf{0} \\ & \mathbf{g}_{43}(\mathbf{X}_3,\mathbf{y}_{3j}) \leq \mathbf{0} \end{aligned} \tag{6.12}$$

6.3 CO Results

Using the same design parameters as were used in the MDF solution, a design solution was obtained using the CO formulation described previously. The CO solution did have difficulty converging, and exhibited sensitivity to the starting design. The first attempt utilized the starting point from Table 6.3, and failed to converge. Other points approximately as far from the optimal solution also failed to converge. As with other implementations in this thesis the system functions were scaled such that they were of the same magnitude. Scaling alone was not adequate to produce convergence— adjustment of the SQP algorithm parameters was required, and the starting design point had to be selected carefully. The successful starting point was $\mathbf{X} = \{0.03, 0.03, 0.03, 0.005, 0.003\}^T$. Table 6.4 displays the design variable and deflection results of the CO solution implementation.

The resulting design and system responses were nearly identical to the MDF results. However, MDF converged in 20 iterations, and CO required 36 system iterations to converge. Each system iteration requires convergence of each subspace

Table 6.4: Results of the SISA CO solution implementation.

design variable	value	response	value
d_1	0.0346	δ_1	0.0270
d_2	0.0348	δ_2	0.0266
d_3	0.0290	δ_3	0.0262
d_{r1}	0.0050	m_{total}	6.956
d_{r2}	0.0030		

optimization, which turned out to be much more expensive than the FPI iterations employed in MDF. One potential difficulty in CO convergence is the difficult attainment of the system level auxiliary equality constraint (a similar constraint is present in IDF and AAO). Other convergence issues with CO have been identified and refinements proposed [1, 14, 24].

This thesis does not endeavor to examine the convergence properties of either ATC or CO, nor does it aim to make any statements concerning what formulation might be better. This thesis does endeavor to explore the distinctions between the solution process of the two strategies. This test problem obviously possessed properties unfavorable to implementation of CO (without formulation refinements). CO generally performs well solving tightly coupled problems, and the test problem is only a moderately coupled problem[4]. Perhaps some of the formulation updates presented in [14] could remedy these convergence problems. Although CO implementation was difficult, the use of the SISA problem was successful since it facilitated a clear illustration of the CO solution process. The difficult convergence of this single example should not be used to gauge the usefulness of CO, which has been demonstrated effectiveness for numerous other design problems.

[4]The SISA problem has 4 of 6 possible couplings, and the strength of these couplings was demonstrated to be moderately strong in the FPI solution— which required 94 iterations.

CHAPTER 7

Analytical Target Cascading Example

The Statically Indeterminate Structural Analysis (SISA) test problem was developed in Chapter 6, and was used to demonstrate the implementation of Collaborative Optimization. In this chapter the same test problem is reformulated into a configuration more well suited for the Analytical Target Cascading (ATC) methodology. This facilitates clear illustration of the distinction between the CO and ATC processes for solving the same design problem.

7.1 Reformulation of SISA Problem for ATC

The ATC methodology is particularly suitable for problems with feedforward coupling, and a unidirectional hierarchical communication structure. The SISA problem was first presented as a non-hierarchic problem with both feedback and feedforward coupling. Rather than using auxiliary constraints to force the problem into a hierarchic form, the original design problem is manipulated algebraically into the desired form. This section presents two possible hierarchic partitioning strategies, outlines the ATC formulation for both partitioning strategies, and summarizes pertinent analysis and design results.

7.1.1 ATC Specific Formulation

The functional relationships required for analysis of the SISA problem were given in equations 6.2 – 6.5. The relations pertinent to deflection analysis for the three-beam case are presented for convenience in equation 7.1 below. Note that the form of some of these relations has been modified slightly. Relations required for stress and mass analysis are omitted for clarity since they are ancillary calculations made once the deflections and forces are known.

$$\delta_3 = \frac{64L^3F_3}{3\pi E d_3^4} \tag{7.1a}$$

$$\delta_2 = \frac{64L^3(F_2 - F_3)}{3\pi E d_2^4} \tag{7.1b}$$

$$\delta_1 = \frac{64L^3(F_1 - F_2)}{3\pi E d_1^4} \tag{7.1c}$$

$$\delta_{r2} = \frac{4F_3 l_{r2}}{\pi E d_{r2}^2} \tag{7.1d}$$

$$\delta_{r1} = \frac{4F_2 l_{r1}}{\pi E d_{r1}^2} \tag{7.1e}$$

$$\delta_2 = \delta_3 + \delta_{r2} \tag{7.1f}$$

$$\delta_1 = \delta_2 + \delta_{r1} \tag{7.1g}$$

One analysis approach is to order the above relationships into a composite function such that a known parameter value can be expressed as a function of a single unknown input. This input is then varied until the resulting parameter value matches the known value. The analysis effectively becomes a root finding problem. If the input value is F_3 and the output of the composite function is $\hat{F}_1$, then the analysis can be stated as:

$$\hat{F}_1(F_3) - F_1 = 0 \tag{7.2}$$

$\hat{F}_1$ is the estimate of the known parameter $F_1 = 1000\ N$, given a guess for the value of F_3. The value of F_3 that satisfies equation 7.2 solves the analysis problem. One possible ordering of the relations in equation 7.1 is shown in Figure 7.1. This flowchart illustrates the functional relationships with arrows and what relations from equation 7.1 are required with encircled letters.

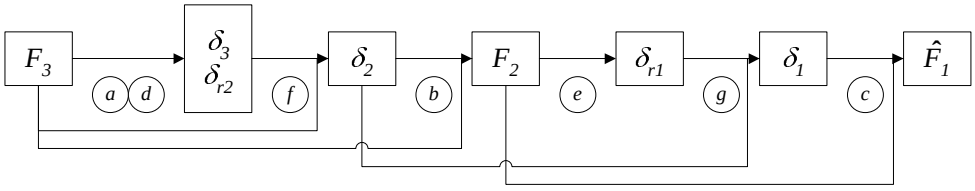

Figure 7.1: Proposed hierarchical analysis sequence of the SISA test problem.

Two different partitioning schemes for this analysis sequence are proposed. The first is a three-level problem, each level having one element. The second is a two level problem, the top level with one element and the lower level with two elements. Both are presented in order to illustrate several important features of ATC. The three-level problem enables the use of an intermediate subspace. Both partitioning schemes together illustrate a wide variety of possible variable handling within the ATC framework.

Partition One

The first partition is sequential in nature, and is illustrated in Figures 7.2 and 7.3. The large boxes in Figure 7.2 represent each of the three subspaces, and are drawn around items that are executed in that subspace. Lines that cross subspace boundaries indicate the required data communication. Figure 7.3 illustrates the partition in the traditional block diagram view. Note that the analysis data is communicated from the bottom up— no feedback is required. In the block diagram coupling variables are input to subspaces via vertical arrows, and design variables are input via

horizontal arrows. During analysis the design variables are of course held constant. It is clear from Figure 7.3 that this partition has no shared variables. Note that in this structure F_3 is neither a response nor a linking variable; it falls outside the Venn diagram of Figure 5.3. F_3 is also not a coupling variable in this partitioning scheme— all that can be said about F_3 is that it is a decision variable for subspace 3 that is an artifact of decomposition. Most applications of ATC have not dealt with such a quantity. The resulting formulation is a new contribution that may lead to application of ATC to a broader range of problems.

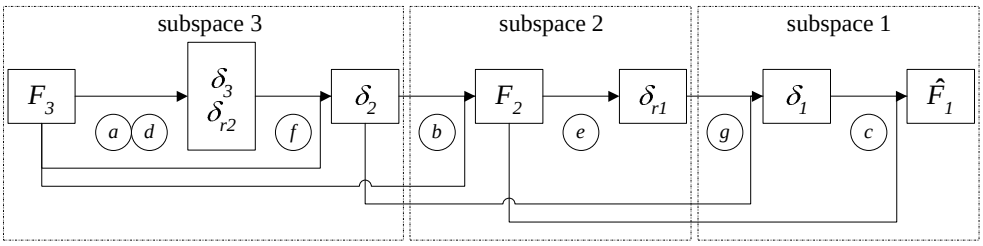

Figure 7.2: First SISA analysis sequence partition.

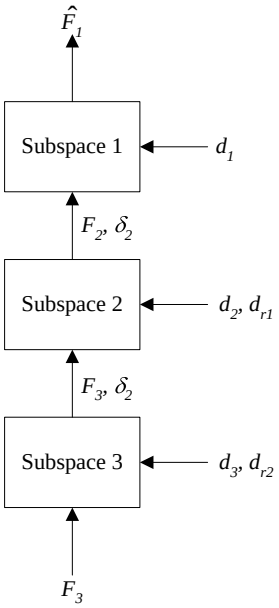

Figure 7.3: First SISA analysis partition diagram.

Partition Two

The second partition illustrates a method for posing the problem as bi-level. This partitioning scheme is illustrated in Figures 7.4 and 7.5, following the same conventions as in the partition one figures. Here F_3 is a linking variable since it is input to both $SS2$ and $SS3$. However, it is not a shared variable since it is not an original design variable, and it is not a coupling variable since it is not passed from one subspace to another. Rather, it is a fabricated decision variable— an artifact of this particular decomposition. The coupling variable δ_2 is neither a response nor a linking variable. The Venn diagram of Figure 5.3 clarifies these categorizations.

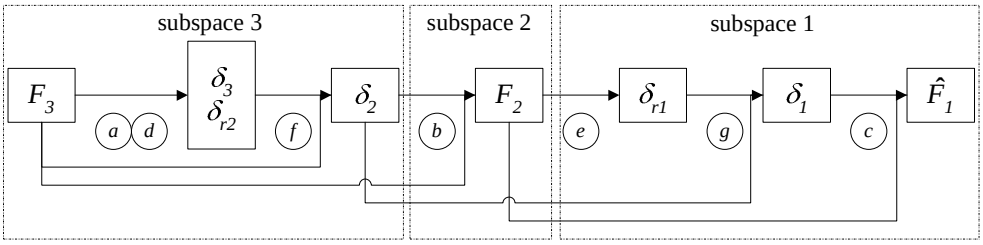

Figure 7.4: Second SISA analysis sequence partition.

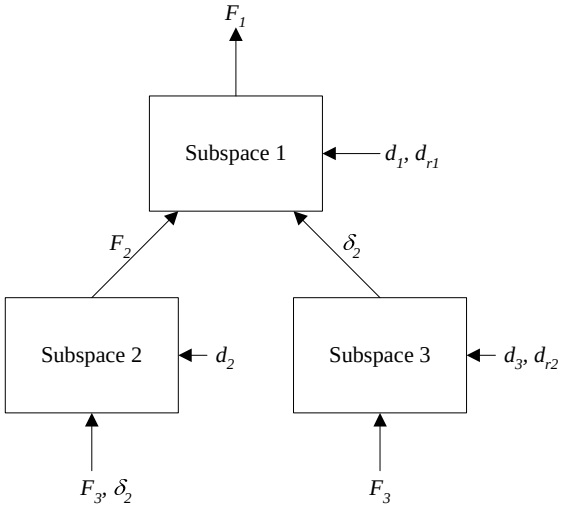

Figure 7.5: Second SISA analysis partition diagram.

7.1.2 Analysis and Baseline Design Results

To establish a baseline analysis and design procedure, equation 7.2 was used to analyze the SISA problem using the design variable and parameter values given in Table 6.3. This root-finding analysis procedure was used in an MDF optimization procedure, where a complete analysis solution is performed at every optimization iteration.

The root finding analysis resulted in the exact same results as the FPI analysis performed in §6.1.3: $\delta_1 = 5.60$ millimeters. The MDF optimization was used to solve the NLP problem of equation 6.1, and resulted in the same results as the MDF and CO approaches implemented in Chapter 6. The parameters of Table 6.3 were used, and stress, mass and transmitted force limits were set to $\sigma_{allow} = 127\ MPa$, $m_{allow} = 7\ kg$, and $Ft_{allow} = 400\ N$. The optimization required twenty iterations to converge, requiring only 5.68 seconds[1]. The optimal design was determined to be:

$$\mathbf{X}_* = \{d_1\ d_2\ d_3\ d_{r1}\ d_{r2}\} = \{.035\ .035\ .029\ .005\ .003\ \}$$

The minimum beam one deflection was $\delta_1^* = 27.0$ millimeters. As with the previous solution strategies in Chapter 6, the mass and transmitted force constraints were active, and none of the stress constraints were active. The resulting design agrees exactly with the MDF and CO approach solutions.

7.2 ATC Formulation

The ATC formulations for both partitioning schemes introduced in the previous section are presented here. This facilitates a more in-depth discussion of the ATC structure, and how design and coupling variables are handled in the ATC formulation.

[1]This indicates that the root-finding approach is the most efficient solution approach for the SISA problem presented thus far.

Partition One

This three level partition illustrates a general ATC formulation in that it possesses an intermediate level that must both coordinate a subspace below and meet targets from above. Subspace 1 is the top level problem, subspace 2 the intermediate level, and subspace 3 the bottom level. Superscripts indicate what subspace the values are computed in.

SS1

$$\begin{aligned} \min_{d_1, F_2, \delta_2, \varepsilon_R} \quad & (\delta_1 - 0)^2 + (F_1 - T_{F_1})^2 + \varepsilon_R \\ \text{subject to} \quad & (F_2 - F_2^{SS2})^2 + (\delta_2 - \delta_2^{SS2})^2 \leq \varepsilon_R \\ & g_{11}(\mathbf{X}) = \sigma_{b1}(\mathbf{X}) - \sigma_{allow} \leq 0 \\ & g_3(\mathbf{X}) = \sum_{i=1}^{3} m_{bi} + \sum_{i=1}^{2} m_{rj} - m_{allow} \leq 0 \\ & g_{41}(\mathbf{X}) = Ft_1 - Ft_{allow} \leq 0 \end{aligned} \tag{7.3}$$

SS2

$$\begin{aligned} \min_{d_2, d_{r1}, F_3, \delta_2, \varepsilon_R} \quad & (F_2 - F_2^{SS1})^2 + (\delta_2 - \delta_2^{SS1})^2 + \varepsilon_R \\ \text{subject to} \quad & (F_3 - F_3^{SS3})^2 + (\delta_2 - \delta_2^{SS3})^2 \leq \varepsilon_R \\ & g_{12}(\mathbf{X}) = \sigma_{b2}(\mathbf{X}) - \sigma_{allow} \leq 0 \\ & g_{21}(\mathbf{X}) = \sigma_{a1}(\mathbf{X}) - \sigma_{allow} \leq 0 \\ & g_{42}(\mathbf{X}) = Ft_2 - Ft_{allow} \leq 0 \end{aligned} \tag{7.4}$$

SS3

$$\begin{aligned} \min_{d_3, d_{r2}, F_3} \quad & (F_3 - F_3^{SS2})^2 + (\delta_2 - \delta_2^{SS2})^2 \\ \text{subject to} \quad & g_{13}(\mathbf{X}) = \sigma_{b3}(\mathbf{X}) - \sigma_{allow} \leq 0 \\ & g_{22}(\mathbf{X}) = \sigma_{a2}(\mathbf{X}) - \sigma_{allow} \leq 0 \\ & g_{43}(\mathbf{X}) = Ft_3 - Ft_{allow} \leq 0 \end{aligned} \tag{7.5}$$

In subspace one the decision variables consist of a local design variable (d_1), two targets for the intermediate problem (F_2 and δ_2), and the response compatibility tolerance (ε_R). No linking variables exist, so ε_y is not required. The target for the original system objective function δ_1 is zero since it is desired to minimize it rather than match a target. The target value for the calculated value of F_1 is T_{F_1}, a known system parameter (the applied force to beam 1).

The subspace two optimization is performed with respect to two local design variables (d_2 and d_{r1}), two targets for the bottom problem (F_3 and δ_2), and the response compatibility tolerance (ε_R). The subspace three optimization is performed with respect to two local design variables (d_3 and d_{r2}), and the uncategorized decision variable (F_3).

Partition Two

The two-level partition lacks an intermediate subspace, but illustrates how a parent problem coordinates two child problems. This formulation also results in a shared variable, allowing the demonstration of a linking variable compatibility constraint and the corresponding tolerance ε_y. Subspace 1 is the top-level problem and subspaces 2 and 3 are the bottom level problems.

SS1

$$
\begin{aligned}
\min_{d_1,d_{r1},F_2,F_3,\delta_2,\varepsilon_R,\varepsilon_y} \quad & (\delta_1 - 0)^2 + (F_1 - T_{F_1})^2 + \varepsilon_R + \varepsilon_y \\
\text{subject to} \quad & (F_2 - F_2^{SS2})^2 + (\delta_2 - \delta_2^{SS3})^2 \leq \varepsilon_R \\
& (F_3 - F_3^{SS2})^2 + (F_3 - F_3^{SS3})^2 \leq \varepsilon_y \\
& g_{11}(\mathbf{X}) = \sigma_{b1}(\mathbf{X}) - \sigma_{allow} \leq 0 \\
& g_{21}(\mathbf{X}) = \sigma_{a1}(\mathbf{X}) - \sigma_{allow} \leq 0 \\
& g_3(\mathbf{X}) = \sum_{i=1}^{3} m_{bi} + \sum_{i=1}^{2} m_{rj} - m_{allow} \leq 0 \\
& g_{41}(\mathbf{X}) = Ft_1 - Ft_{allow} \leq 0
\end{aligned}
\tag{7.6}
$$

SS2

$$\begin{aligned} \min_{d_2,F_3} \quad & (F_2 - F_2^{SS1})^2 + (F_3 - F_3^{SS1})^2 \\ \text{subject to} \quad & g_{12}(\mathbf{X}) = \sigma_{b2}(\mathbf{X}) - \sigma_{allow} \leq 0 \\ & g_{42}(\mathbf{X}) = Ft_2 - Ft_{allow} \leq 0 \end{aligned} \tag{7.7}$$

SS3

$$\begin{aligned} \min_{d_3,d_{r2},F_3} \quad & (F_3 - F_3^{SS1})^2 + (\delta_2 - \delta_2^{SS1})^2 \\ \text{subject to} \quad & g_{13}(\mathbf{X}) = \sigma_{b3}(\mathbf{X}) - \sigma_{allow} \leq 0 \\ & g_{22}(\mathbf{X}) = \sigma_{a2}(\mathbf{X}) - \sigma_{allow} \leq 0 \\ & g_{43}(\mathbf{X}) = Ft_3 - Ft_{allow} \leq 0 \end{aligned} \tag{7.8}$$

The subspace 1 optimization is performed with respect to two local variables (d_1 andd_{r1}), two targets for bottom level responses (F_2 and δ_2), a linking variable target (F_3), and both the response and linking variable compatibility tolerances (ε_R and ε_y). The subspace 2 optimization is performed with respect to a local design variable (d_2), and a linking variable (F_3). The subspace 3 optimization is performed with respect to two local design variables (d_3 and d_{r2}), and a linking variable (F_3).

This formulation has a curious property regarding the δ_2 term. It is set as a target by the top-level problem, and matched by the subspace 3 response. However, it is also required as an input for the subspace 2 analysis. δ_2 is neither a decision variable nor a target, yet its value changes throughout the ATC process. The solution to this issue is to pass the current δ_2 target value from the top level problem to subspace 2 and treat it as a design parameter.

7.3 ATC results

The ATC process was used to solve the SISA test problem using the second partitioning strategy and the same design parameter values used for the other SISA

problem solutions (Sections 6.1.3, 6.3, and 7.1.2). The compatibility constraint tolerances both went to zero (indicating a consistent system), and the ATC process converged to essentially the same solution as the other solution methods. The stopping criterion was set to a relatively large value here to reduce run time, so the level of precision in the results is slightly reduced. The minimum beam 1 deflection was found to be $\delta_1 = 27.1$ millimeters, and the optimal design was determined to be:

$$\mathbf{X}_* = \{d_1\ d_2\ d_3\ d_{r1}\ d_{r2}\} = \{.035\ .033\ .031\ .003\ .002\ \}$$

The SISA test problem was effective at illustrating two different applications of the ATC methodology. Reformulation of the test problem into a hierarchic structure allowed a more natural implementation of ATC than would the formulation from Chapter 6. The two different hierarchic partitioning schemes facilitated illustration of most of the major quantity types present in the ATC methodology. A final comparison between the CO and ATC methods in provided in Chapter 8.

CHAPTER 8

Conclusion

This thesis introduced the topic of complex system optimization, showed how it relates to product development, and provided an overview of necessary preliminary topics. Strategies for complex system optimization are motivated by needs of industries concerned with the design of complex products. A critical review of three important single-level formulations and two selected multi-level formulations was presented. This review illustrated how some of the said industry needs, such as reduced design time and improved product performance, are addressed by the formulations considered in the review. Two new engineering design examples were fully developed and used to demonstrate four of the five covered formulations. This chapter summarizes the findings from the critical review, and poses open questions and suggestions for future work.

8.1 Single-Level MDO Formulations

Single-level formulations employ a single optimizer at the top level of a system. Three single-level formulations were reviewed: Multidisciplinary Feasible (MDF), Individual Disciplinary Feasible (IDF), and All-at-Once (AAO). In single-level formulations, the system optimizer makes all design decisions; single-level strategies

are said to have centralized design. However, in each case investigated, the system analysis is partitioned into smaller analysis subspaces— these formulations have distributed analysis.

The most basic and most commonly used formulation is MDF. The MDF formulation differs little from the standard design problem. A complete system analysis is performed for every optimization iteration— from the optimizer's perspective MDF is no different from a standard optimization problem. MDF requires the use of some type of system analysis tool. Fixed Point Iteration is used regularly as a system analysis tool. It was shown that this algorithm can have convergence difficulties, which affect the performance of the MDF process. Using an optimization algorithm to aid the analysis process can help alleviate these problems.

In the IDF formulation the system optimizer assists the analysis process by providing estimates for coupling variables during the system optimization. IDF performs analysis and design simultaneously. The dimension of the system design problem is increased, but the subspace analysis can be executed in parallel (reducing design cycle time). The AAO formulation shifts even more responsibility to the system optimizer by requiring it to provide estimates for the state variable values. AAO performs analysis, design, and evaluation of governing equations simultaneously. The dimension of the system design problem is further increased, but AAO can still provide a highly efficient solution method.

The design of a turbine blade for a gas-turbine engine was used to illustrate the MDF and IDF formulations. The analysis of the turbine blade was posed as a coupled system and partitioned into thermal and structural analyses. Both MDF and IDF were successful at solving this design problem. The turbine blade analysis was equipped with means to vary coupling strength. It was shown that while the

MDF approach required substantially greater computation time for strongly coupled problems, the computation time required by IDF did not increase.

All single-level formulations have centralized decision making. This does not map well to some existing organizations composed of design teams led by disciplinary experts, or to organizations with existing specialized optimization tools. Multi-level formulations address this issue by employing subspace optimizers, providing means for distributed analysis *and* design.

8.2 ATC and CO Comparison

Two multi-level formulations were covered in detail: Analytical Target Cascading (ATC), and Collaborative Optimization (CO). Each had distinct origins, and are linked to different industries: ATC to the automotive industry, and CO to the aerospace industry. The nature of a methodology is colored by its origin; implicit emphases are inseparably connected to original intent. CO was developed based on Multidisciplinary Analysis needs and under a by-discipline partitioning paradigm. A natural result is an implicit emphasis on analysis. ATC stemmed from needs to formalize product development in a hierarchically structured organization, aligned by object. ATC is characteristically focused on physical aspects of a product.

Although CO and ATC had contrasting origins and were intended for different purposes, the resulting mathematical formulations seem similar. Each formulation could possibly be extended to fit problems natural for the other. For example, additional constraints could be imposed on a general non-hierarchic problems in order to fit the ATC structure. Similarly, a problem with a multi-level hierarchic structure could be forced into a bi-level problem for solution with CO. Mathematically these extensions are feasible, but would practical application prove successful? Would the

transformed problems behave well? Intuitively it does not make sense to force an multidisciplinary problem into a multilevel, ATC type structure— there is no reason to put disciplinary analyses on different levels. Alternatively, forcing a multilevel structure into a bi-level problem in order to implement CO requires breaking natural communication pathways. For example, in an automotive design team it makes sense for an engine design group to report its results to the powertrain design group. A forced CO formulation would require all design groups to report directly to the top level instead.

Some extension beyond a methodology's original purpose seems reasonable. For example, ATC was designed as an early product development tool. However, using detailed simulations permits its use as a strategy for the entire design process. Additionally, could CO be used to set targets for product development? This and other questions about other potential extensions of ATC and CO deserve further investigation.

Another result of the different contexts these methodologies were developed in is the way in which quantities are classified. In most cases the same quantities exist in each formulation; they are however classified differently. These classification schemes were juxtaposed in Chapter 5, and clarified using a Venn diagram and illustrated with the example of Chapters 6 and 7. For convenience the terminology definitions are presented here as well. Of particular importance is the distinction between shared variables and linking variables.

Three key terms used in MDO formulations are:

$\mathbf{x}$ *Local design variables*: Design variables that are each inputs to only one subspace.

$\mathbf{x}_s$ *Shared design variables*: Design variables that are each inputs to more than one

subspace.

$\mathbf{y}$ *Coupling variables*: Quantities that are passed from one subspace to another that are not original design variables, but rather artifacts of decomposition.

Four important terms for ATC formulations are listed below. The † symbol is used to distinguish the ATC terms when a symbol already used for MDO nomenclature is employed.

$\mathbf{y}^\dagger$ *Linking variables*: Quantities that are input to more than one subspace. These could be either shared variables (original design variables) or coupling variables (artifacts of decomposition— not original design variables).

$\mathbf{x}^\dagger$ *Local decision variables*: Variables that a particular subspace determines the value of. May or may not be original design variables.

$\mathbf{R}$ *Responses*: Values generated by subspaces required as inputs to respective parent subspaces. May or may not be coupling variables.

$\mathbf{T}$ *Targets*: Values set by parent subspaces to be matched by the corresponding quantities from child subspaces. Targets may exist for either responses or linking variables.

Note that coupling variables may be either linking variables or responses, and shared variables are always linking variables. Quantities may also exist that are neither shared nor coupling variables, but could be linking variables or responses.

Table 8.1 summarizes several properties of the ATC and CO methodologies.

Table 8.1: ATC and CO comparison summary.

Property	ATC	CO
▷Process configuration	Nested coordination	Nested optimization
▷Number of top-level optimizations	Multiple	Single
▷Numerical	Proven convergence	
▷Intended usage	Early product development	Multidisciplinary Analysis and product design
▷Auxiliary constraints	Inequality w/ tolerance	Equality
▷Targets	Linking variables & responses	Shared & coupling variables
▷Quantity classification	$\mathbf{x}^{\dagger}$, $\mathbf{y}^{\dagger}$, $\mathbf{R}$, $\mathbf{T}$	$\mathbf{x}$, $\mathbf{x}_s$, $\mathbf{y}$, $\mathbf{z}$, $\hat{\mathbf{z}}$
▷Problem structure	Hierarchic	General non-hierarchic
▷Intended partitioning	Object (example: automotive)	Aspect (example: aerospace)

8.3 Future Work

Several questions were posed in this chapter that merit further investigation. More in-depth studies regarding Analytical Target Cascading and Collaborative Optimization should be conducted to clarify distinctions between the methodologies, and to answer questions such as:

- Should ATC be employed to solve problems with general non-hierarchic structures?
- Can CO be used as a product development target setting tool?
- Can CO be extended from a bi-level to a multilevel formulation?

Some of these questions could be answered by carefully selecting and utilizing test problems to further elucidate the properties of each methodology. One interesting approach could be to nest CO within ATC. Recall that CO has an implicit focus on analysis, while ATC is intended to be a product development tool. One nesting approach could be to use ATC to coordinate the overall system problem (partitioned by object), and use CO to optimize each object (partitioned by discipline).

In addition to a review of CO and ATC, other important methodologies should be considered as well, such as Concurrent Subspace Optimization (CSSO) and Bi-Level Integrated System Synthesis (BLISS). Clear elucidation of these formulations in a single work would be a remarkable contribution, providing means for identifying what formulations best suit particular problems, and for assessing what needs have yet to be met.

8.4 Final Remarks

Strategies for complex system optimization have profound potential for improving product quality and performance and for reducing design cycle time. These strategies facilitate understanding and exploiting synergistic interactions between system members, and provide means for task parallelization, while frequently allowing use of existing expertise and resources.

A greater understanding of the spectrum of available strategies will encourage better utilization in industry. This thesis provides a basic exposition of selected fundamental strategies. No single formulation can be said to be superior; each has strengths and fills a specific niche. Some methodologies have been developed and refined in different industries— in parallel and in relative independence. Collaboration among the proponents of these separate methodologies could lead to synergistic results.

An understanding of the many available approaches to complex system optimization is an advantage to practitioners seeking the best solution to their own design problem. This understanding is enhanced by sharing information and experience, particularly across industry boundaries. A vision is to have available the tools necessary to effectively map mathematical formulations to general complex system optimization problems in industry, particularly beyond the few industries currently engaged in these activities. This thesis is a small step towards this end.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Natalia M. Alexandrov and Robert Michael Lewis. Analytical and computational aspects of collaborative optimization. Technical Report NASA/TM-2000-210104, NASA, April 2000.

[2] Eyal Arian. Convergence estimates for multidisciplinary analysis and optimization. Technical Report ICASE Report No. 97-57, NASA, October 1997.

[3] R. J. Balling and C. A. Wilkinson. Execution of multidisciplinary design optimization approaches on common test problems. *AIAA Journal*, 35:178–186, 1997.

[4] Richard J. Balling. Approaches to MDO which support disciplinary autonomy. In *Multidisciplinary Design Optimization - State of the Art, Proceedings of the ICASE/NASA Langely Workshop on Multidisciplinary Design Optimization*. SIAM, 1997.

[5] Richard J. Balling and J. Sobieszczanski-Sobieski. Optimization of coupled systems: a critical overview of approaches. In *5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analyis and Optimization*. AIAA, 1994.

[6] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley and Sons, Inc., second edition, 1993.

[7] Ashok D. Belegundu and Tirupathi R. Chandrupatla. *Optimization Concepts and Applications in Engineering*. Prentice Hall, New Jersey, 1999.

[8] Boeing. Desired attributes of an engineer. *http://www.boeing.com/companyoffices/pwu/attributes/attributes.html*. Accessed November 20, 2004.

[9] R. D. Braun and I.M. Kroo. Development and application of the collaborative optimization architecture in a multidisciplinary design environment. In *Multidisciplinary Design Optimization - State of the Art, Proceedings of the ICASE/NASA Langely Workshop on Multidisciplinary Design Optimization*. SIAM, 1997.

[10] Robert D. Braun. *Collaborative Optimization: An Architecture For Large-Scale Distributed Design*. PhD thesis, Stanford University, April 1996.

[11] Steven C. Chapra and Raymond P. Canale. *Numerical Methods for Engineers*. McGraw-Hill, third edition, 1998.

[12] Baihong Chen, Jihong Liu, Yifang Zhong, and Ji Zhou. Properties of the coupled factors in MDO and their application in collaborative optimization. In *Proceedings of DETC'00, ASME 2000 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. ASME, 2000.

[13] Evin J. Cramer, J. E. Dennis Jr., Paul D. Frank, Robert Michael Lewis, and Gregory R. Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal of Optimization*, 4:754–776, 1994.

[14] Angel-Victor DeMiguel and Walter Murray. An analysis of collaborative optimization methods. In *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. AIAA, 2000.

[15] K. English, C. L. Bloebaum, and E. Miller. Development of multiple cycle coupling suspension in the the optimization of complex systems. *Structural and Multidisciplinary Optimization*, 22:268–283, 2001.

[16] R. Fellini, H.M. Kim, M. Kokkolaras, N. Michelena, and P.Y Papalambros. Target cascading for design of product families. In *Proceedings of the 4th Congress on Structural and Multidisciplinary Optimization*, June 4-8 2001.

[17] American Institure for Aeronautics and Astronautics Inc. (AIAA). Current state of the art in multidisciplinary design optimization. Technical report, MDO Technical Committee, January 1991.

[18] Joseph P. Giesing and Jean-François M. Barthelemy. A summary of industry MDO applications and needs. *AIAA*, 1998.

[19] A. A. Giunta, V. Balabanov, S. Burgee, B. Grossman, R. T. Haftka, W. H. Mason, and L. T. Watson. Variable-complexity multidisciplinary design optimization using parallel computers. *Computational Mechanics '95— Theory and Applications*, pages 489–495, 1995.

[20] P. Hajela, C. L. Bloebaum, and J. Sobieszczanski-Sobieski. Application of global sensitivity equations in multidisciplinary aircraft synthesis. *Journal of Aircraft*, 27:1002–1010, 1990.

[21] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[22] Donald R. Jones. DIRECT. In *Encyclopedia of Optimization*. Kluwer Academic Publishers, 1999.

[23] H.M. Kim. *Target Cascading in Optimal System Design*. PhD thesis, University of Michigan, 2001.

[24] Hyung Min Kim, Nestor F. Michelena, Panos Y. Papalambros, and Tao Jiang. Target cascading in optimal system design. *Journal of Mechanical Design*, 125:474–480, September 2003.

[25] CS. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, May.

[26] S. Kodiyalam and J. Sobieszczanski-Sobieski. Multidisciplinary design optimization - some formal methods, framework requirements, and application to vehicle design. *Int. J. Vehicle Design (Special Issue)*, pages 3–22, 2001.

[27] Ramprasad Srinivasan Krishnamachari. *A Decomposition Synthesis Methodology For Optimal Systems Design*. PhD thesis, University of Michigan, 1996.

[28] I.M. Kroo. Mdo for large scale design. In *Multidisciplinary Design Optimization - State of the Art, Proceedings of the ICASE/NASA Langely Workshop on Multidisciplinary Design Optimization*. SIAM, 1997.

[29] William H. Mason, Zafer Gürdal, and Raphael T. Haftka. Experience in multidisciplinary design education. In *ASEE Annual Conference*, June 1995.

[30] N.F. Michelena, H.A. Park, and P.Y. Papalambros. Convergence properties of analytical target cascading. In *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. AIAA-2002-5506, 2002. Atlanta, GA.

[31] John Robert Olds. *Multidisciplinary Design Techniques Applied to Conceptual Aerospace Vehicle Design*. PhD thesis, North Carolina State University, 1993.

[32] P.Y. Papalambros and D.J. Wilde. *Principles of Optimal Design: Modeling and Computation*. Cambridge University Press, New York, second edition, 2000.

[33] M.J.D. Powell. Algoritms for nonliear constraints that use lagrangian functions. *Mathematical Programming*, 14:224–248, 1978.

[34] Peter J. Röhl, Beichang He, and Peter M. Finnigan. A collaborative optimization environment for turbine engine development. Technical Report AIAA-98-4734, General Electric Corporate Research and Development, 1998.

[35] J. Sobieszczanski-Sobieski and R. T. Haftka. Multi-disciplinary and multi-level optimum design. *Computer Aided Optimal Design: Structural and Mechanical Systems*, 1987.

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

[37] Jaroslaw Sobieszczanski-Sobieski. Optimization by decomposition: A step from hierarchic to non-hierarchic systems. In *Second NASA/USAF Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*. NASA, 1988.

[38] Jaroslaw Sobieszczanski-Sobieski. Sensitivity of complex, internally coupled systems. *AIAA Journal*, 28, 1990.

[39] Jaroslaw Sobieszczanski-Sobieski, Jeremy S. Agte, and Robert R. Jr. Sandusky. Bilevel integrated system synthesis. *AIAA Journal*, 38:164–172, 2000.

[40] Jaroslaw Sobieszczanski-Sobieski, Troy D. Altus, Matthew Phillips, and Robert Sandusky. Bi-level integrated system synthesis (BLISS) for concurrent and distributed processing. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. AIAA, 2002.

[41] Rangarajan K. Sundaram. *A First Course in Optimization Theory*. Cambridge University Press, 1996.

[42] R Tappeta, S Nagendra, J.E. Renaud, and K Badhrinath. Concurrent sub-space optimization (CSSO) code usage in iSIGHT. Technical Report 97CRD188, GE, January 1998.

[43] Karl T. Ulrich and Steven D. Eppinger. *Product Design and Development*. McGraw-Hill, second edition, 2000.

[44] Terrance C. Wagner. *A General Decomposition Methodology For Optimal System Design*. PhD thesis, University of Michigan, 1993.

[45] C.F. Wu and Michael Hamada. *Experiments: Planning, Analysis, and Parameter Design Optimization*. John Wiley and Sons, Inc., 2000.