THE DESIGN OF A SIMULATION-BASED FRAMEWORK FOR THE DEVELOPMENT OF SOLUTION APPROACHES IN MULTIDISCIPLINARY DESIGN OPTIMIZATION

by

Kevin F. Hulme

A dissertation submitted to the
Faculty of the Graduate School of
State University of New York at Buffalo in partial
fulfillment of the requirements for the degree of
Doctor of Philosophy

January 20, 2000

Acknowledgments

First and foremost, I would like to thank Dr. Christina L. Bloebaum for introducing me to the field of MDO, and for more than five years of support, guidance, advisement, and friendship.

I would next like to thank my parents, family, and friends for their constant support and understanding.

I would further like to thank my committee members, Dr. Kemper Lewis, Dr. Roger Mayne, and Dr. Rakesh Nagi, for their respective inputs to this dissertation.

I would also like to recognize my colleagues in the MODEL laboratory, in particular, Ken English, Eliot Winer, Collin McCulley, Bryan Moulton, Chen-Hung Huang, Mohamad Kasim Abdul-Jalil, Omar Conteh, and Yuji Nozaki. It has been a privilege to work alongside each of you.

Last, but not least – a tip of the hat to *lunchie*.

Table of Contents

List of Tables

List of Figures

Abstract

A primary goal of Multidisciplinary Design Optimization (MDO) is to decompose a large multidisciplinary system into a related grouping of smaller, more tractable, coupled subsystems. Often, the resulting decomposition is not fully hierarchical in nature and thus requires iterative techniques to attain a converged system analysis and associated optimal design solution. The optimal design of such multidisciplinary systems as automobiles and aircraft might require hundreds or even thousands of iterative cycles to attain numerical convergence. As might be expected, this high level of iteration is both timely – the process might take months or years for a true multidisciplinary system – and computationally costly. Broadly speaking, the goal of the present research is to increase the efficiency associated with the design of large-scale, multidisciplinary engineering systems. To accomplish this task, the current work contributes four areas of unique research to the MDO community.

The first research area presents the design and continual development of the CASCADE simulation tool, which generates analytical representations of coupled multidisciplinary design problems corresponding to both the system analysis and the optimization portions of a large-scale multidisciplinary design. Due to the lack of availability of real-world multidisciplinary design data, there is a research need for a capability to simulate the coupling structure and behavior of a decomposed engineering system. CASCADE suits this research need - its simulations can be useful for testing a variety of new tools and technologies in MDO.

The second area of research contribution involves the development of a new heuristic means for the convergence of a multidisciplinary analysis, called the Data Fusion Analysis Convergence (DFAC) algorithm. This algorithm utilizes a neural network scheme to model the input-output behavior of each subsystem output quantity. Thereafter, gradient-based optimization is used to correct the errors in each of the neurons, concurrently. As a means for coordination, a data fusion-based approach is then implemented to intelligently blend together discrepant information resulting from the error minimization process. Thereafter, a new estimate for each subsystem output is formed, and the entire process repeats until convergence. The DFAC algorithm builds upon the strengths of two well-known formal means for analysis convergence, Fixed-point Iteration (FPI) and Newton's Method (NM). Through preliminary testing using CASCADE-based simulation, DFAC has shown itself to be more efficient than FPI in all simulations thus far, and more reliable than NM, which tends to diverge in situations where little is known about the starting solution point.

The third area of research contribution involves a large-scale comparison of three popular means for posing and solving the entire MDO design cycle, the Multiple-Discipline Feasible (MDF), All-At-Once (AAO), and Individual-Discipline Feasible (IDF) approaches. A large-scale comparison of these strategies had not been possible prior to this time, due to the deficiency of test problems in the MDO community. The presence of the CASCADE simulator obviates this deficiency. Each strategy varies in how it treats the system analysis and optimization portions of the MDO design cycle. Initial results have shown that MDF is the most reliable strategy for attaining the greatest overall design improvement, but at the largest associated cost, by far. AAO usually

attains substantial design improvement at a much lower cost, and IDF typically attains a solution whose characteristics are intermediate to these two extremes. The disparity in performance between the solution strategies tends to increase with both problem size and nonlinearity.

The fourth and final area of research contribution presents the development of a computational MDO framework, entitled FACETS (Framework for the Analysis of Coupled Engineering Techniques in Simulation), which provides designers with an all-encompassing computational infrastructure. FACETS contains a multitude of MDO tools and techniques intended for large-scale coupled system reduction. The ultimate purpose of FACETS is to provide a preliminary design tool that can allow a design manager to identify potential means for time and cost reduction within the elaborate multidisciplinary design process, in a simulation-based setting. At present, the feature modules of FACETS are the CASCADE simulator, a solution strategies (MDF/IDF/AAO) module, and an analysis convergence (FPI/NM/DFAC) module. In so doing, FACETS brings together the other three areas of research contribution into a single MDO environment. FACETS also includes an optimization module, a system planning module, and an elaborate post-processor for result verification.

Chapter 1

Introduction

Large-scale engineering design is a constantly evolving discipline. Design managers are consistently trying to identify means for producing a “better” product in a “shorter” period of time. The design of aircraft, spacecraft, automobiles, and others demands a methodology that is more sophisticated and efficient than a traditional, serial design approach. This approach is characterized by a sequential design cycle, with respect to the participating design groups. A design is formulated in a given design group and passed to the next group, who uses the previous groups’ output variables as input variables. This “over the wall” design approach is intuitive to implement, but lacks the sophistication, ingenuity, and efficiency required by the high-technology designs of the forthcoming century. For this reason, the serial design approach is slowly becoming obsolete in favor of a methodology known as Concurrent Engineering [11,31].

Concurrent Engineering is a systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support. The interaction of all participating engineering groups throughout a large-scale engineering design cycle is a truly *multidisciplinary* effort. In the early 1980's, a field of research

emerged which inherently attempts to unite the concepts of Concurrent Engineering with large-scale, multidisciplinary (coupled) engineering design. This emerging field has since been coined Multidisciplinary Design Optimization, or MDO [74]. The general MDO approach is intuitive: divide a single large task into a grouping of smaller, interrelated (coupled), and more manageable sub-tasks. The large task is often referred to as a *system*, and the smaller, interrelated tasks are often referred to as *subsystems*. Each subsystem typically contains *design variables*, which are parameters that might change during a formal optimization procedure. In addition, each subsystem typically also contains additional unknown outputs, often referred to as *behavior variables*. It is these variables that might change during a complex system analysis. Further, these variables represent the coupling links between the subsystems.

Advantageous as the MDO approach might seem for engineering design, it does contain numerous inherent drawbacks. Namely, the decomposition of the system rarely yields a fully hierarchical breakdown of disciplinary subsystems. This means that the design cycle must necessarily be iterative, which will add time and cost to the process. Hence, it is the goal of many MDO researchers to identify means for ***reducing the time and cost associated with the multidisciplinary design cycle***. Broadly speaking, this is the ultimate goal of the present research effort.

To accomplish this goal, the present research effort focuses on four specific areas of MDO research. The first and most vital area is that of system simulation. The development of new tools and techniques in MDO requires a safe and robust means for testing the new technology, prior to its implementation on a "real world" product or process. Analytical means have been devised for simulating the structure and semantics

of a multidisciplinary system. Such test systems can then be used for conducting meaningful research relating to increasing the efficiency of MDO. The recent work of McCulley [54,55], for instance, utilizes simulation-based procedures to determine the optimum sequence of a series of input/output black boxes or "modules." Such information might be useful for determining the most efficient strategy to numerically converge a system of nonlinear equations [56], for example. Such a procedure can take place either sequentially or in parallel.

The second area of MDO research in this effort involves the complicated task of posing and subsequently solving large-scale engineering design problems. During the late 1980's and early 1990's, numerous researchers [26,27,14,15,3] have developed both "conventional" and "alternative" solution strategies for accomplishing this very task. Past studies have been limited to very simple application problems, mainly due to the deficiency of test problems in the MDO community. Newly developed simulation tools allow present-day researchers to overcome this deficiency, as is the case in the present research.

The third area of MDO research in this effort focuses on the most costly aspect of the multidisciplinary design cycle - the iterative system analysis. Because of the lack of hierarchy that typically stems from the system decomposition, iteration becomes a necessity. There exist numerous formal means for accomplishing this task, each of which has associated strengths and weaknesses. In response to this, a heuristic convergence methodology has been developed. This methodology hopes to provide a robust and efficient alternative to formal convergence techniques in certain situations.

Finally, the fourth area of research in this effort discusses the development of an all-encompassing computational MDO framework. This framework embodies all of the aforementioned research concepts, and has been designed in an open-ended, modular form so as to accommodate future MDO technologies as well. It is the goal of this framework to provide the MDO design manager with a tool that can be used to build a simulation-based representation of a true "real world" system. Thereafter, a variety of solution techniques can be efficiently explored, which hopes to provide the user with insight on how to proceed with the design and analysis of the true system.

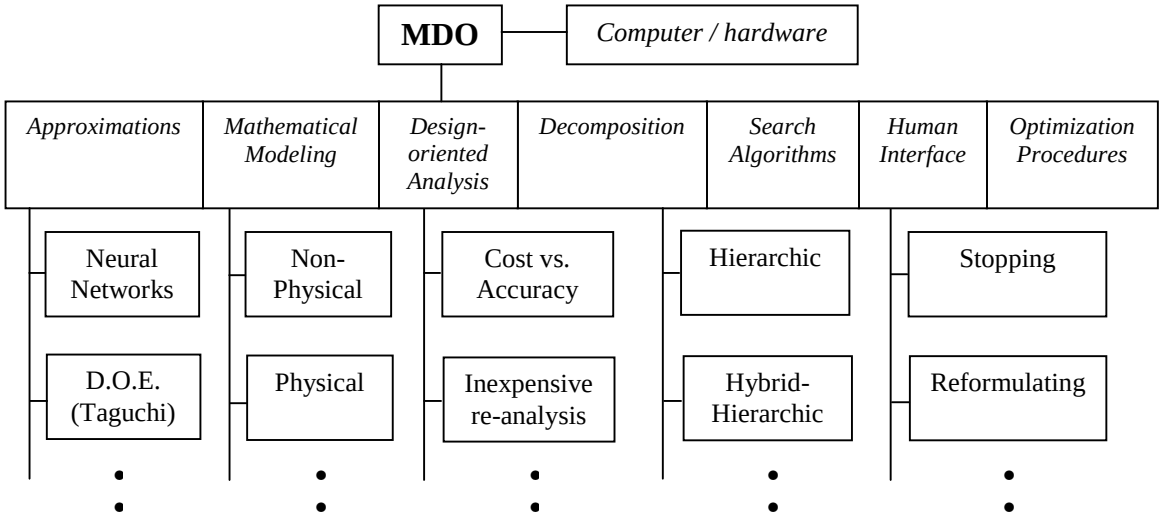


Figure 1.1: MDO principal conceptual components and their breakdowns

The present research effort aims to address numerous research areas within the field of MDO. The Figure 1.1 flowchart taken from [78] demonstrates the principal conceptual components of Multidisciplinary Design Optimization, and their partial breakdowns into more specified areas of research. It is clear that the four primary areas of research addressed by the present research effort together embody a portion of all 7 sectors in this flowchart. Namely, the system simulation research encapsulates the

Mathematical Modeling and *Decomposition* sectors, while the solution strategies research encapsulates the *Design-oriented analysis* and *Optimization Procedures* sectors. Further, the analysis convergence research encapsulates the *Approximations* and *Search Algorithms* sectors, while the framework development research encompasses the *Human Interface* sector, among others.

Chapter 2 presents an in-depth background discussion on the field of Multidisciplinary Design Optimization. In so doing, this chapter explores the related works of past researchers in MDO, provides a motivation for conducting the present research, and concludes with a specific statement of the research goals of this work.

Chapter 3 presents a discussion of the continual design and development of CASCADE [37,38,39], a multidisciplinary system simulation tool. The capabilities, features, and robustness of this tool have necessarily been expanded for use in the present research effort. The chapter begins with a formal discussion of past research efforts relating to system simulation and benchmark MDO test problems.

Chapter 4 discusses numerous common and alternative approaches for posing and subsequently solving multidisciplinary design problems. After a formal presentation of these previously developed strategies, the CASCADE simulator is used to explore and compare these solution strategies to an extent that had not before been possible.

Chapter 5 presents a discussion of analysis convergence. Formal convergence techniques are presented initially, along with their inherent strengths and weaknesses. In response to these, a new and heuristic convergence alternative technique is presented, and demonstrated on numerous classes of test problems.

Chapter 6 discusses the continual development of FACETS [42], a computational framework tool that encompasses all of the research concepts presented in this work. FACETS provides an environment for simulating large-scale multidisciplinary design problems, and allows the user to explore numerous techniques and methods for solution. This chapter is preempted with a lengthy discussion of existing frameworks and problem solving environments in the field of MDO.

Finally, Chapter 7 summarizes this research work, presents some concluding remarks, and recommends numerous avenues for future work.

Chapter 2

Background and Motivation

Fundamental to the field of Multidisciplinary Design Optimization (MDO) is the discipline of numerical optimization itself. A brief discussion of the terminology and the mathematics behind a standard optimization problem is presented first. Thereafter, an in-depth discussion of MDO and its many focal areas of research is offered, a majority of which relate directly to the present research effort. This background presentation serves to bring the motivation of the present research to the forefront, which is formally stated at the end of this chapter.

Optimization

Optimization is a concept that has significance in most of our everyday lives. The term “optimize” makes one think of similar terms such as “enhance”, “improve”, and “increase”. In the context of engineering, we typically wish to “produce the best quality of life, using the resources that are available” [84].

A formally stated optimization problem has numerous parameters. The *objective function* is the cost function that is being either minimized or maximized. The *design variables* are changeable parameters that signify a potential for change. The *constraints* are limitations on the design space. Constraints can be of numerous forms, including

equality, inequality, and side constraints. A typical optimization problem has the mathematical form of Equation 2.1.

Minimize:	$F(\mathbf{X})$		objective function	
Subject to:	$g_j(\mathbf{X}) \leq 0$	$j = 1, m$	inequality constraints	
	$h_k(\mathbf{X}) = 0$	$k = 1, l$	equality constraints	**[2.1]**
	$X_i^l \leq X_i \leq X_i^u$	$i = 1, n$	side constraints	
where:	$\mathbf{X} = \{X_1 \ X_2 \ X_3 \ .. \ X_n\}$		design variables	

With this general understanding of the discipline of optimization, the specialized field of Multidisciplinary Design Optimization is now investigated in detail. This discussion begins with the topic of System Decomposition.

System Decomposition

In the early 1980's, Sobieski laid the foundations for a field that is now known as Multidisciplinary Design Optimization, or MDO [74]. The fundamental objective of MDO is to develop an improved design capability, while considering disciplinary interactions for synergistic affects. The underlying principle of MDO is quite simple: divide a large task into a sequence of smaller, interrelated (coupled) and more individually manageable tasks. In engineering optimization problems, the large task is commonly known as a *system*, and the smaller, interrelated tasks are known as *subsystems*. Each subsystem contains independent input quantities in the form of *design variables*, as well as additional dependent input quantities that are actually output quantities from other subsystems. These are commonly referred to as *behavior variables*. Refer to Figure 2.1. Shown in subsystem Y, which generates a single output Y_1, and is a function of two inputs, X and W.

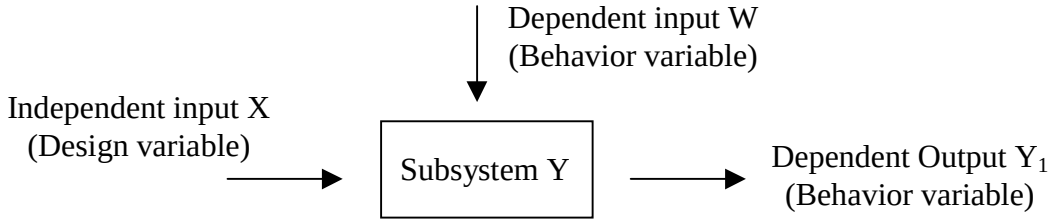


Figure 2.1: Subsystem input types

Note that in an MDO context, the objective and constraint functions will likely be a function of both the system design variables and the subsystem outputs, which are typically referred to as behavior variables. Hence, it might be beneficial to restate Equation [2.1] accordingly, as shown in Equation [2.2].

Minimize:	$F(\mathbf{X}, \mathbf{Y})$		objective function	
Subject to:	$g_j(\mathbf{X}, \mathbf{Y}) \le 0$	$j = 1, m$	inequality constraints	
	$h_k(\mathbf{X}, \mathbf{Y}) = 0$	$k = 1, l$	equality constraints	
	$X_i^l \le X_i \le X_i^u$	$i = 1, n$	side constraints	**[2.2]**
	$Y_i^l \le Y_i \le Y_i^u$	$i = 1, p$	side constraints	
where:	$\mathbf{X} = \{X_1\ X_2\ X_3\ ..\ X_n\}$		design variables	
	$\mathbf{Y} = \{Y_1\ Y_2\ Y_3\ ..\ Y_p\}$		behavior variables	

To accomplish the division of the system, Sobieski applied a linear decomposition method [74] for large-scale multidisciplinary problems. This methodology is applicable to hierarchic (top-down) systems, such as the one depicted in Figure 2.2. In such a hierarchic system, there is a definite ordering to the execution of each module to produce a final and exact result. A non-hierarchic system contains lateral (two-way) couplings, which essentially means that the system has no discernable “starting point”. Between many of the modules in a non-hierarchic system, there exist two-way couplings. In other

words, the output of one module is the input to a second module, and vice-versa (refer to Figure 2.3, which exhibits lateral coupling between subsystems Y and Z).

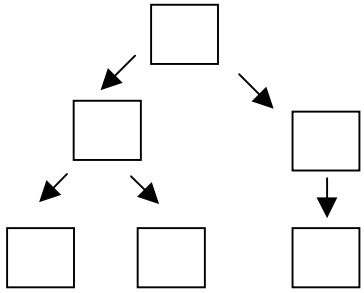


Figure 2.2: Hierarchic system model

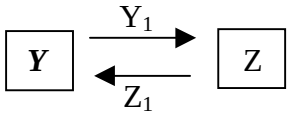


Figure 2.3: Lateral couplings

The system analysis for a non-hierarchic system requires an initial guess to the magnitude of each output module, and subsequent iteration to gain convergence. Most decomposed engineering systems exhibit traits from both hierarchic and non-hierarchic systems, and are called hybrid-hierarchic, as seen in Figure 2.4.

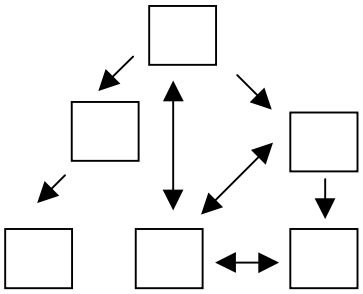


Figure 2.4: Hybrid-hierarchic system model

The Global Sensitivity Equation (GSE) method was the first to successfully extend Sobieski's concept of hierarchic modularity to non-hierarchic systems, as early as 1988 [76,77].

Sensitivity Analysis - Global Sensitivity Equations

A sensitivity is defined as a change in an output value, with respect to a given input value. System sensitivities are required in optimization to gain system improvement by prescribing a change in the subsystem design variables. A sensitivity analysis can be a computationally costly procedure, and must therefore be efficient. The Global Sensitivity Equation (GSE) approach defines the total derivatives of the output quantities in terms of *local sensitivities*. These local sensitivities are partial derivatives of each subsystem's outputs with respect to its inputs. For continuous functions, local sensitivities are computed analytically. For complex functions, numerical procedures such as finite difference methods [72] are often required to attain these sensitivities.

To illustrate the mathematics of the GSE method, consider the two subsystem schematic seen in Figure 2.5. Subsystem A could be thought of as the structures discipline, and subsystem B, the aerodynamics discipline, of the design of an automobile, for example. Subsystem A has two sets of inputs: design variables X_A, and the output (coupling) from subsystem B, Y_B . Similarly, subsystem B has design variables X_B and the output (coupling) from subsystem A, Y_A, serving as its inputs.

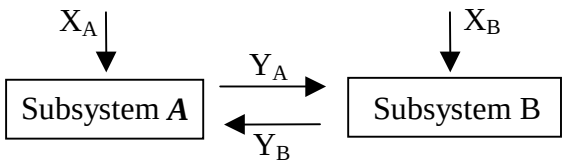


Figure 2.5: Two-subsystem interaction

This interaction can be expressed as follows:

$$Y_A = f(X_A, Y_B)$$
$$Y_B = f(X_B, Y_A) \quad \textbf{[2.3]}$$

Expanding equation [2.3] with a first order Taylor series gives:

$$\frac{dY_A}{dX_A} = \frac{\partial Y_A}{\partial X_A} + \frac{\partial Y_A}{\partial Y_B} * \frac{dY_B}{dX_A}$$
$$\frac{dY_B}{dX_B} = \frac{\partial Y_B}{\partial X_B} + \frac{\partial Y_B}{\partial Y_A} * \frac{dY_A}{dX_B} \quad \textbf{[2.4]}$$

Applying the chain rule to equation [2.4] gives:

$$\frac{dY_A}{dX_B} = \frac{\partial Y_A}{\partial Y_B} * \frac{dY_B}{dX_B}$$
$$\frac{dY_B}{dX_A} = \frac{\partial Y_B}{\partial Y_A} * \frac{dY_A}{dX_A} \quad \textbf{[2.5]}$$

Finally, expressing equations [2.4] and [2.5] in matrix form yields:

$$\begin{bmatrix} 1 & -\frac{\partial Y_A}{\partial Y_B} \\ -\frac{\partial Y_B}{\partial Y_A} & 1 \end{bmatrix} \begin{bmatrix} \frac{dY_A}{dX_A} & \frac{dY_A}{dX_B} \\ \frac{dY_B}{dX_A} & \frac{dY_B}{dX_B} \end{bmatrix} = \begin{bmatrix} \frac{\partial Y_A}{\partial X_A} & 0 \\ 0 & \frac{\partial Y_B}{\partial X_B} \end{bmatrix} \quad \textbf{[2.6]}$$

The leftmost square matrix on the left side of equation [2.6] is known as the Global Sensitivity Matrix (GSM) and is comprised of the couplings between interacting subsystems. In other words, the GSM is a matrix of the partial derivatives of all "output" behavior variables with respect to all "input" behavior variables. The dimension of this matrix is (p x p), where p is the total number of behavior variables in the system. The matrix on the right hand side of the equation is a matrix of partial sensitivities of all behavior variables with respect to all system design variables. The dimension of this

matrix is (p x n), where n is the total number of system design variables. The center matrix (the rightmost matrix on the left-hand side of the equation) is the desired matrix of total derivatives. These derivatives provide an indication of how a change in one or more design variables will affect **all** of the behavior variable outputs of the system.

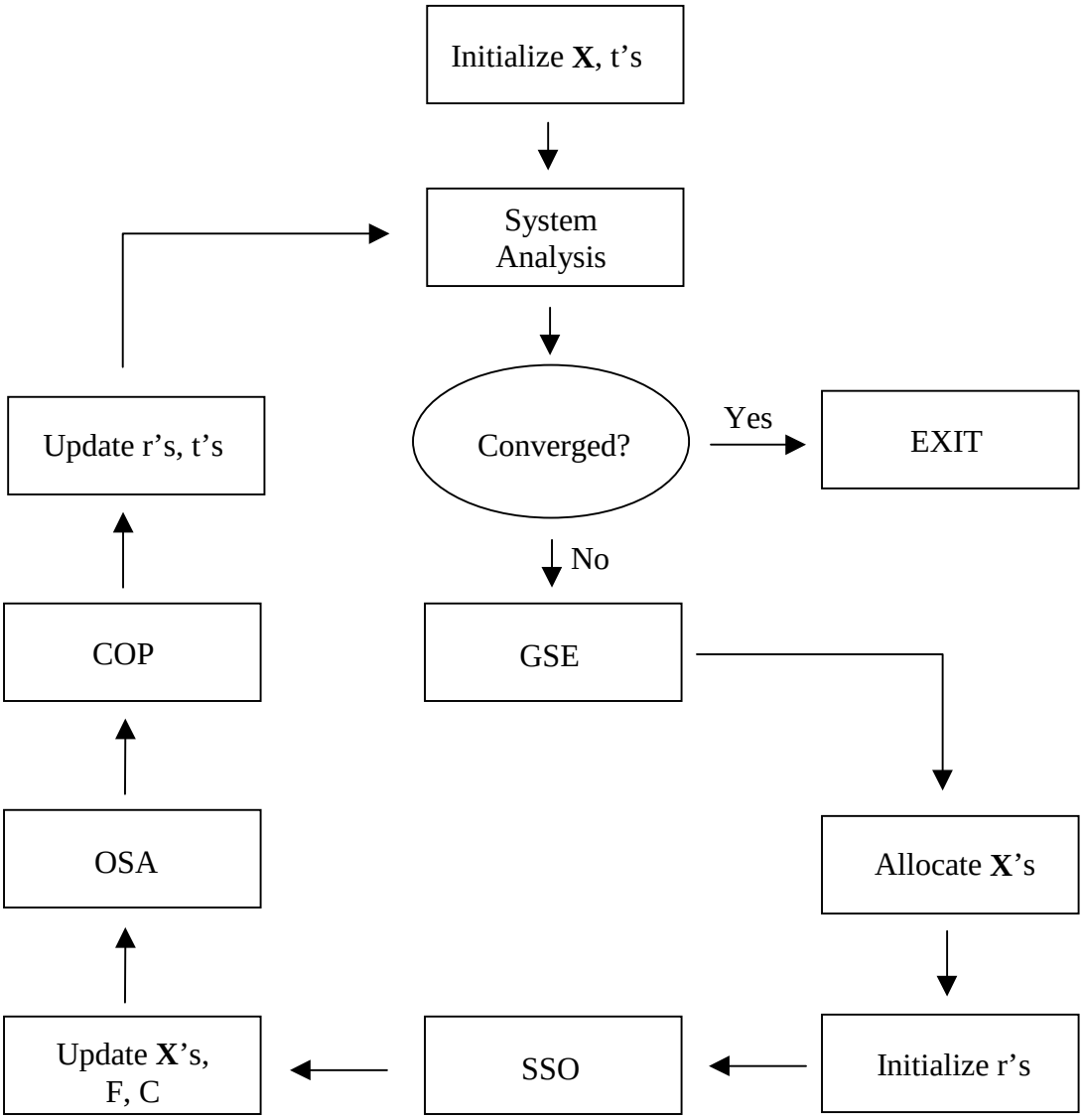


Figure 2.6: Concurrent Subspace Optimization flowchart

The Concurrent Subspace Optimization (CSSO) method [75,6] takes the concept of the GSE method one step farther. Refer to Figure 2.6. Both the sensitivity analyses and the optimizations are performed at the subsystem level subsequent to the decomposition of a large engineering system. During the CSSO procedure, design variables **X** are allocated to the *subspace* upon which they are found to have the greatest impact. (Note: A "subspace" is typically associated with the optimization portion of a decomposed multidisciplinary system, while a subsystem is typically associated with the analysis portion. The subspace and subsystem typically do not identically coincide). This determination is made based upon the comparison of sensitivities of the system objective function F with respect to a cumulative constraint function C, by means of a K.S. function [30], for each subspace. Subsequent to design variable allocation, temporarily decoupled optimizations (SSO's) are performed in each subspace concurrently. Realize that design variables will likely have an impact on many of the other subspaces to which they were not allocated. Due to this decoupled nature of these concurrent optimizations, a means for coordination between the subspaces, commonly called the coordination optimization procedure (COP), must be performed thereafter. An optimum sensitivity analysis (OSA) must be performed prior to this coordination procedure. The OSA is based on the Kuhn-Tucker conditions [90], and provides sensitivity information of the system objective function F, with respect to the parameters used for coordination, which are typically denoted r and t. This overall cycle repeats until convergence.

With the given background on engineering optimization, system decomposition, and sensitivity analysis, the design synthesis for generic hybrid-hierarchic multidisciplinary problems is presented.

Conventional Multidisciplinary Design Synthesis

The methodology for conventional multidisciplinary design synthesis is illustrated in Figure 2.7.

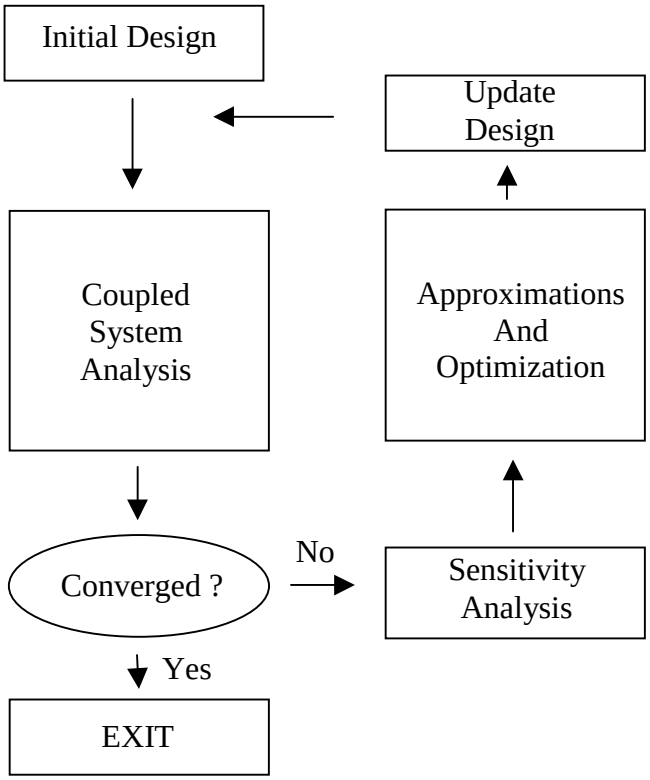


Figure 2.7: Hybrid-hierarchic design synthesis (MDF)

The overall procedure is explained as follows. With a given set of design variables, the system outputs (behavior variables) are initialized, which allows for an initial system analysis involving all the decomposed subsystems. It is during the system analysis that a converged set for the behavior variables **Y** is sought. Because the system decomposition is assumed to have been hybrid-hierarchic in nature, the system analysis is highly

iterative in nature. Note that during this procedure, the design variables **X** are held constant. Analysis convergence is checked thereafter. If the system has not converged, sensitivities are computed. The sensitivities are then used within a gradient-based optimizer to improve the design by perturbing the system design variables. This updated design is once again fed to the system analyzer. The process repeats itself until both the design and behavior variables have achieved a desired level of convergence.

The procedure that has just been described is the most frequently implemented and most intuitive approach for posing and solving design problems in MDO. Today, this design cycle is commonly referred to as the Multiple-Discipline Feasible [14,15,3] (MDF) solution strategy. This and other multidisciplinary solution strategies will be discussed in greater detail in Chapter 4. For now, it is important for the reader to note the potential for extremely high cycle time and computational cost associated with the MDF strategy. One can see how a great deal of resources might be expended during numerous phases of he MDF cycle, namely the sensitivity analysis and especially the nonlinear and highly iterative system analysis. The next section discusses an area of research that is related to increasing the efficiency of the costly system analysis.

Dependency Structure Matrix, Task Scheduling and Convergence

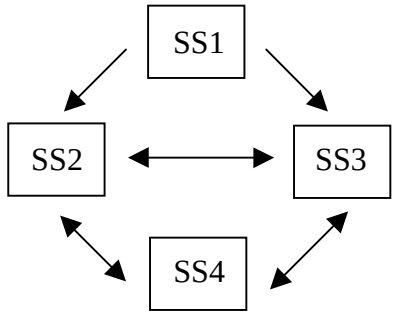


Figure 2.8: Four subsystem hybrid-hierarchic interaction

Consider the four subsystem hybrid-hierarchic decomposed system of Figure 2.8. This coupled system can be represented as a square Dependency Structure Matrix (DSM), (formerly called the *Design* Structure Matrix), wherein each of the subsystems is denoted as a box along the diagonal [80]. Figure 2.9a depicts a randomly ordered DSM for the system of Figure 2.8.

Figure 2.9: a. Randomly oriented DSM; b. Optimally oriented DSM

To understand this figure fully, one must refer to Figure 2.8. It is clear that subsystem 3 requires input information from subsystems 1, 2, and 4 and transmits output information to subsystems 2 and 4. Now note the ordering of Figure 2.9a. This information is translated to that figure as follows: subsystem 3 receives information in the form of feedforwards from subsystems 4 and 2, and receives information in the form of a feedback from subsystem 1. Similarly, subsystem 3 passes information to subsystems 2 and 4 in the form of feedbacks. From this example, it is clear that feedbacks represent pieces of information, commonly referred to as "couplings", that are required from subsystems that are located *downstream* in the solution process. As a result, an initial guess and an iterative framework are required to converge a system with feedbacks. Feedbacks are the filled black squares located below the diagonal in Figure 2.9, and represent a coupling link between two subsystems. An optimally arranged system would

have 100% feedforwards, which are filled back squares located above the diagonal in Figure 2.9. Such a system ordering would not require any iteration whatsoever.

One area of interest within the field of MDO is known as task sequencing [54,55]. Optimal sequencing of a system is the result of reordering the sequence of boxes in the DSM so as to maximize the efficiency of the iterative system analysis. This is accomplished primarily by reducing the DSM feedbacks. Figure 2.9b is an illustration of an optimally ordered DSM representation of the Figure 2.8 system. Here, the number of feedbacks has been reduced from 5 to 3. This reduction will reduce the number of analysis iterations required for system convergence on each MDF solution cycle.

During a multidisciplinary design cycle, a great deal of time and effort is often expended in attempting to converge a large system of behavior variables. Task sequencing has been found to have a profound impact on an inter-related field of research - convergence strategy development. McCulley [55,56] defines a convergence strategy as being "made up of rules which determine, based on the current state of the system and its particular structure or content, what module or modules to execute." McCulley has developed both sequential and parallel convergence strategies that are based on parameters such as: modular update requirements, computational resource limitations, and prioritization of modular characteristics. He has also found that in addition to sequencing, other factors that affect multidisciplinary convergence include both the semantics and the degree of parallelism of the system.

Another benefit of task sequencing is to aid in identifying weak links. Once a system has been optimally sequenced, the sensitivities of the couplings can be analyzed. Those that are found to be comparatively weak can be suspended for a pre-determined

number of iterations of the convergence procedure, or eliminated outright. Rogers' DeMAID tool [67,68] has been in development for many years and has been used successfully for task sequencing, as well as (amongst numerous other features) the identification of subsystem couplings found to be comparatively strong, and others found to be comparatively weak. The process of identifying weak couplings is an important area of research, but not a trivial one. Recent findings in this very sector of MDO are presented next.

Coupling Suspension

The system designer needs to know the relative strength of all of the couplings between disciplinary subsystems. This knowledge may allow for the temporary suspension or the outright elimination of select couplings. Such could provide a less costly and more computationally efficient iterative system analysis.

Bloebaum's method for assessing coupling strength is based on local sensitivities [7]. This method finds the numerical value of the sensitivity of every coupling using the previously explained GSE method. These sensitivities are normalized, and then compared. Modules that are found to have the smallest set of normalized sensitivities are considered to be weak couplings. These couplings are then considered for temporary suspension or outright elimination, providing that they do not have a substantial impact further downstream. This determination is made based on downstream coupling strengths.

Recall that many optimization problems have objective functions and constraint functions that *have outputs of complex systems -- behavior variables -- as their optimization variables*. Hence, the drawback of the local sensitivity method for

identifying weak couplings is that it does not relate the local coupling strengths to their effect on the global problem, namely the objective function and constraints. A coupling may have a relatively small normalized sensitivity, but may still have a large effect on the objective function or constraints. This coupling may therefore have a major impact on the accuracy of the design solution.

More recently, researchers have been looking for ways to assess coupling strengths based on global sensitivities. English, Bloebaum, and Miller have developed such a method [16,17]. Their analysis creates a separate coupling strength for the objective function and each constraint. This work suggests that temporary suspension of certain couplings only occur in staggered intervals, due to the difficulty in predicting which constraints will be active during any given system analysis iteration. In addition, this work suggests that the criterion for the total elimination of a coupling be based on a user-defined percentage of the magnitude of the objective function.

Having discussed a great many of the areas of research within the field of MDO, a motivation for the present research is now realized. In addition, a statement of the specific goals that this research attempts to achieve is formally presented.

Motivation and Research Goals

Based on the background discussion presented thus far, which has dealt with numerous specified areas of research in the field of MDO, one particular point should be made clear. Namely, the primary goal of a majority of MDO research efforts is to simplify or shorten the design process in any of a number of ways, without appreciably sacrificing accuracy. In so doing, there is a great potential for savings in both design time and cost (financial, computational, and otherwise). The present research is no exception.

To address this broadly-stated research goal, a number of previous efforts have focused on the development of problem-solving computational environments, commonly called frameworks, that are used for the solution process of some or all portions of a multidisciplinary design problem. Many of the recently developed MDO frameworks share a number of common strengths, but also suffer from a similar set of deficiencies. Namely, there appears to be a need for a framework that can exploit many of the newly developed tools, strategies and techniques in MDO. Such techniques include optimization methods, approaches for sensitivity analysis, MDO solution strategies, task sequencing strategies, coupling strength assessment, analysis convergence strategies, and many others not explicitly presented in this background discussion.

An inter-related problem is the lack of availability of design data and benchmark (test) problems. Researchers must have a safe and robust means for testing a newly developed strategy prior to its implementation on an actual "real-world" design. Hence, the development of a framework that is focused on the incorporation of new MDO tools and techniques should have a robust means for coupled system simulation, both at the system analysis and optimization levels, as its foundation.

This research provides an original contribution to the field of MDO by providing a new means, in the form of a simulation-based computational framework, for aide in the reduction of time and cost of the multidisciplinary design process. To accomplish this ambitious objective, four specific research goals of the present research are stated as follows:

- First, the continual design and development of the simulation tool, called CASCADE, that is used to provide the multidisciplinary problem data for use within the

framework. This module provides the design manager a means for constructing a robust analytical representation of a coupled multidisciplinary system. The system will have a known structure, but unknown (randomly generated) coupling characteristics, hereafter referred to as "semantics". Because the framework is simulation-based, CASCADE serves as the flagship module of the present research effort.

- Second, the design and development of a framework module that focuses on means for structuring and subsequently solving all aspects (system analysis, sensitivity analysis, and optimization) of a multidisciplinary design problem. This module is referred to at the *Solution Strategies* module throughout this research effort. This module provides the design manager with means for solution alternative to that depicted in Figure 2.7, which is the conventional MDF solution procedure.
- Third, the design and development of a framework module that focuses solely on the most costly aspect of the multidisciplinary design cycle -- the system analysis. The *Analysis Convergence* module provides the design manager with numerous means for approaching the often highly iterative analysis that is associated with a multidisciplinary design. Options within this module include formal convergence means, and newly developed heuristic convergence means.
- Finally, the design and development of the computational framework tool itself, entitled FACETS, which serves as a modular aide for exploring numerous techniques in multidisciplinary design. The FACETS framework embodies each of the other three primary accomplishments of the present research, bulleted above. FACETS serves as a *preliminary* tool for a design manager, to be used on a simulated

representation of a real-world multidisciplinary system. Through the use of simulation, FACETS hopes to provide insight to a design manager as to the appropriateness of certain MDO techniques and strategies given a system with known decomposition structure, but unknown semantic characteristics. This insight can then aid a design manager in choosing the "best" overall solution approach for the true multidisciplinary system.

An entire chapter shall be devoted to each of these four areas of specified research. Chapter 3 begins with a presentation on multidisciplinary system simulation, and focuses specifically on the development of the CASCADE simulation tool.

Chapter 3

Complex System Simulation

The present research makes extensive use of the CASCADE simulator for generating analytical representations of real-world multidisciplinary systems. The initial design of the CASCADE simulator began in 1994 and was coded using the Fortran 77 [36] computer language. This "baseline design" was completed in January of 1996 [37]. The simulator was re-coded in ANSI C [47,81] in 1997 for numerous reasons. The most important reason was the need for dynamically allocating storage space for the great many multi-dimensional arrays required to store the system data. This chapter presents an overview of the design of the simulator, followed by an in-depth discussion of how the simulator has been expanded to accommodate the needs of the present research work. Before doing so, a brief literature survey of past-recent work relating to multidisciplinary system simulation is presented.

Literature Survey

The initial motivation for the design of CASCADE stemmed from late 1980's research that has been conducted at NASA Langley Research Center. Padula and Young [61] first developed a computer simulator designed to simulate and improve multilevel optimization research. Their effort utilized simple analytic functions to represent

complex engineering analyses in hopes of testing multilevel decomposition strategies in a short period of time. A later work of Padula and Sobieski [62] furthered the application of the simulator. For this work, the simulator was used for experimentation with a large variety of candidate algorithms for multilevel design optimization methodologies.

Numerous research efforts pertaining to complex system simulation have appeared subsequent to the design of the initial CASCADE simulator in 1994 and 1995. The first of these also involves NASA Langley, specifically, their Test Suite for benchmark MDO problems [63]. The test suite is a World Wide Web-based platform for collecting, distributing and maintaining standard test problems in the field of MDO. The test suite houses 3 classes of benchmark problems, ranging from simple Class I problems to challenging Class III problems which generally have no known global solution. While the test suite attempts to address the lack of commonly accepted and readily available benchmark problems in the MDO community, many of it problems are difficult to implement and/or manipulate for specified research needs.

A recent research effort pertaining to system simulation was published by Balling and Wilkinson in 1997 [4]. This research work utilizes a class of synthetic problems for testing MDO approaches, including single-level optimization, CSSO approaches, and Collaborative Optimization (CO) [50] approaches. The test systems are created so as to emulate the general form of numerous seminal equations in structural optimization, such as the deflection and the bending of a beam. This adds to the realism of the semantics of the simulations. The test problems that are generated are simple, analytical, convex functions where known (exact) solutions are possible. This is advantageous in that the success of an MDO method can be assessed and compared quickly and easily. However,

these test problems may prove insufficient for the simulation of highly complex and nonlinear systems with “real world” characteristics.

The CASCADE simulator hopes to build upon the strengths and sidestep the known weaknesses of these and other means for system simulation.

CASCADE - System Construction

CASCADE has been designed to allow the creation of analytical test systems of user-prescribed size and structure, and randomly generated semantic characteristics. This allows a design manager to generate a test system that has a coupling structure that is similar to that of a “real world” system. This test system, however, will have behavioral traits that are initially unknown to the design manager. In this way, a design manager can generate numerous “instances” of a multidisciplinary system with fixed structural characteristics, but randomly varying semantic characteristics. Using these test systems will thus allow the design manager to safely assess the appropriateness of a given MDO solution method over a wide range of possible system behavior. The knowledge and insight gained from such simulations can then be used towards the design of the true engineering system.

Prior to the execution of CASCADE to generate and converge a system of analysis equations, the user must assign numerous input options. The user assigns a variety of options, including the number of subsystems, design variables, and behavior variables in the system, which is assumed to be hybrid-hierarchically decomposed. The user assigns numerous additional options, including an initial seed value for random number generation, the number of design variables per subsystem, and convergence

characteristics. Once these preliminaries are handled, the system of analysis equations is constructed, term by term.

The terms that are generated and combined to create the coupled set of analysis equations are of polynomial nature, and take the following general form:

$$y_k = \sum_{i=1}^{n} a_i x_i^{e_i} + \sum_{j=1}^{m} b_j y_j^{f_j}, \quad k = 1, p$$

[3.1]

Here, behavior variable y_p is the p^{th} output quantity of subsystem Y. In this example case, subsystem Y has a total of k output quantities. This output is a function of two types of input terms. The first type are independent input quantities x, which are the subsystem design variables. Each of the n design variable terms has an associated coefficient, a, which can be either positive or negative. Each design variable term also contains an exponent e, which can presently take on one of three values: 1, 2, or 3. These design variables remain constant during an iterative system analysis. The second type are dependent input quantities y, which represent outputs (behavior variables) from the other subsystems in the decomposed system. Like the design variable terms, each of the m behavior variable terms has an associated coefficient, b, which can be either positive or negative, and an exponent f, which can presently take on one of the same three values. Refer to Figure 3.1 for an illustration.

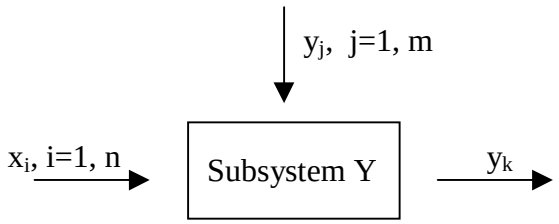


Figure 3.1: Generic coupling depiction for subsystem Y

The system of equations is constructed as follows. First, the semantic nature of every term in every behavior variable of the system is determined, as illustrated in equation [3.1]. For each term in the system, the sign, coefficient and exponent are randomly determined. (The magnitude of the randomly generated coefficient is likely altered soon thereafter to guarantee a stable set of equations. This procedure is described in detail shortly). Again, these properties constitute the semantics of the generated system. The structure of the system; the decision as to which behavior variables serve as inputs to each output behavior variable is pre-determined either randomly or by user-specification, and stored in a boolean matrix. In essence, this matrix houses the couplings of the system. A "1" represents a coupling between two behavior variables, and a "0" signifies no coupling between two behavior variables. For instance, the design manager might input the boolean matrix seen in equation [3.2] to represent the coupling structure viewed in Figure 3.2.

$$\begin{matrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{matrix} \qquad \textbf{[3.2]}$$

Here, subsystem 1 receives input information from subsystems 2 and 3, but does not receive input from subsystem 4. This is ascertained from the 1's and 0's seen in the first

column of the boolean matrix. Similarly, subsystem 2 receives input only from subsystem 4, subsystem 3 receives input from subsystems 1 and 4, and subsystem 4 receives input from subsystem 1,2, and 3.

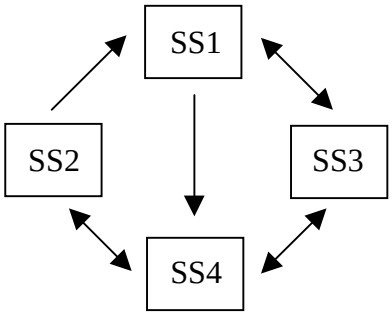


Figure 3.2: Example input coupling structure

Note that stability is a large issue when generating an arbitrary set of nonlinear equations. To generate a set of equations that are to be well behaved within some set of pre-determined bounds, one must incorporate some heuristic knowledge a priori. Namely, a range must first be assigned for the maximum and minimum values for the final values of each of the behavior variable output equations. This has been determined to be ± 9999.0, somewhat arbitrarily. (This quantity is referred to as the *maximum equation magnitude (MEM)* in the discussion that follows). Given this knowledge up front, the terms in the system can be generated "safely" as described in the following paragraph.

For each behavior variable in the system, the maximum magnitude is divided by the total number of terms (both design and behavior variable) in that equation. This quantity is designated as being the *maximum term magnitude (MTM)* for the given behavior variable equation. Hence, all terms in a given behavior variable equation y_k must have a coefficient b_j whose magnitude obeys the following:

$$b_j (MEM)^{f_j} \leq MTM$$

[3.3]

Hence:

$$b_j \leq \frac{(MTM)^{f_j - 1}}{MEM}$$

[3.4]

If the initially generated term coefficient is found to be larger than this threshold value, then a new coefficient is randomly generated between 0 and this upper limit.

Having constructed a set of equations that are known to be of sufficient stability, CASCADE then proceeds to identify a converged solution. This is accomplished by initializing all subsystem design variables and all subsystem outputs (behavior variables) to a randomly generated value between ± 9999. During the iterative system analysis, the values of the design variables remain fixed. It is the converged solution to the behavior variables that is sought. CASCADE uses and easy and robust technique to achieve the initial converged solution, a procedure called Fixed-point Iteration [45]. This and other means for numerical convergence of coupled equations will be formally discussed in Chapter 5.

Upon identifying a converged solution, CASCADE generates numerous output files pertaining to the system representation. These include a data file that stores, among other parameters, the design variable magnitudes, and the initialized and converged behavior variable magnitudes. Also generated is a file that contains the analysis equations themselves, written using character strings. The output file is a compilable ANSI C header ".h" file, and can be compiled and used with other computer codes.

To summarize the functionality of CASCADE that has been described thus far, an example system is now presented.

Example System - Analysis Representation

Figure 3.3 consists of a multidisciplinary system that has been hybrid-hierarchically decomposed into four coupled subsystems.

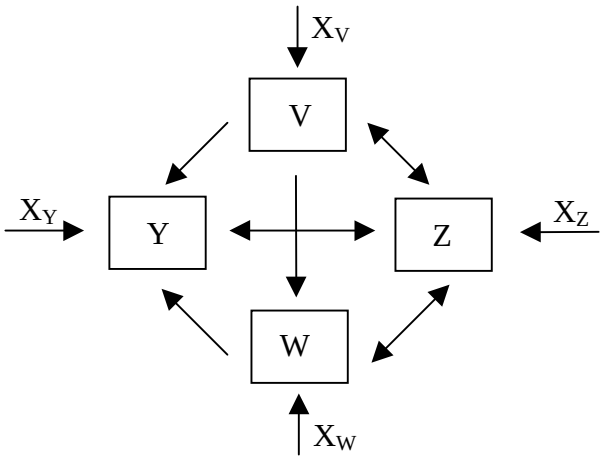


Figure 3.3: Example CASCADE system (system analysis)

For the sake of review, note that the input boolean matrix that is used to generate this coupling structure appears as follows:

	V	**W**	**Y**	**Z**
V	0	1	1	1
W	0	0	1	1
Y	0	0	0	1
Z	1	1	1	0

[3.5]

Each subsystem has its own set of independent design variables as input (the “X’s”), as well as dependent behavior variables - outputs from other subsystems - also serving as input (the “V’s”, “W’s”, “Y’s”, and “Z’s”). One possible set of CASCADE-generated

analysis equations that exhibit the coupled behavior illustrated in Figure 3.3 might appear as in equation [3.6]:

$$V = 0.22X_V^1 + 4.5E\text{-}9Z^3$$
$$W = -9.5E\text{-}6X_W^2 - 8.6E\text{-}6V^2 + 5.6E\text{-}10Z^3$$
$$Y = 3.6E\text{-}6X_Y^2 + 0.19W^1 + 4.1E\text{-}6V^2 + 0.03Z^1$$
$$Z = 8.0E\text{-}10X_Z^3 - 0.05V^1 + 0.11W^1 - 4.7E\text{-}10Y^3$$

[3.6]

Once the equations are generated, the system can be converged by first initializing the variables. Table 3.1 summarizes the initialized variable values for this test system.

Table 3.1: CASCADE test system - initial values

X_V	811.71
X_W	-9054.2
X_Y	2112.0
X_Z	5150.7
V	666.6
W	1984.23
Y	-69.69
Z	312.92

Upon substituting these values into equation [3.6], the following "calculated" values for behavior variables V, W, Y, and Z result after the first iteration:

Table 3.2: First iteration results

V	**W**	**Y**	**Z**
178.7140	-782.6004	-123.1175	294.2527

These "calculated" values are the new initial guess for the next iteration. The iterative process repeats until all variables converge to within some pre-specified criterion. This is numerical process is the aforementioned Fixed-point Iteration convergence procedure. For this example, an absolute convergence criterion of 0.01 will be enforced for all 4

behavior variable values. The overall convergence results from the given starting point are summarized in Table 3.3.

Table 3.3: Overall convergence results

	V	W	Y	Z
0	666.6	1984.23	-69.69	312.92
1	178.7140	-782.6004	-123.1175	294.2527
2	178.6908	-779.0565	-123.0042	14.2963
3	178.5762	-779.0707	-131.4058	14.6873
4	178.5762	-779.0703	-131.3940	14.6917
5	178.5762	-779.0703	-131.3939	14.6917

Parallelization of system analysis

As discussed in Chapter 2, the decomposition of an engineering system will typically result in a hybrid-hierarchic structure. This form of decomposition, where the system variables and equations are decomposed, can be more specifically referred to as *model decomposition* [18]. The goals of model decomposition are to reduce problem sizes, provide a more convenient problem definition, and use specialized solution methods within the subsystems. Similarly, the implementation of parallel processing can be thought of as *computational decomposition* [18]. The benefit of this form of decomposition is primarily a reduction in computational time. Very often, computational decomposition is used in combination with model decomposition, where the independent subproblems are solved in parallel.

This analysis convergence procedure is slightly modified to operate in a distributed computing environment, via Parallel Virtual Machine, or PVM [21]. The PVM system uses message passing to allow programmers to exploit distributed computing, using any of a wide variety of computer types. A key concept in PVM is that

it makes a collection of individual computers (referred to as *hosts*) appear as one large *virtual machine.*

A master machine enrolls in PVM, and performs preliminaries such as reading of the input parameters and array declaration. The master then spawns slave tasks on each of the host machines available on the virtual machine. The master then packs and sends array data for one subsystem, to each slave. Each slave receives and unpacks the data sent to it by the master. Thereafter, the slave performs the previously described analysis procedure for the one subsystem that was sent to it. Each slave then sends the newly computed data back to the master for a convergence check. The process repeats until the system of analysis equations converges. Once convergence occurs, a process termination signal is sent to each slave machine (refer to Figure 3.4).

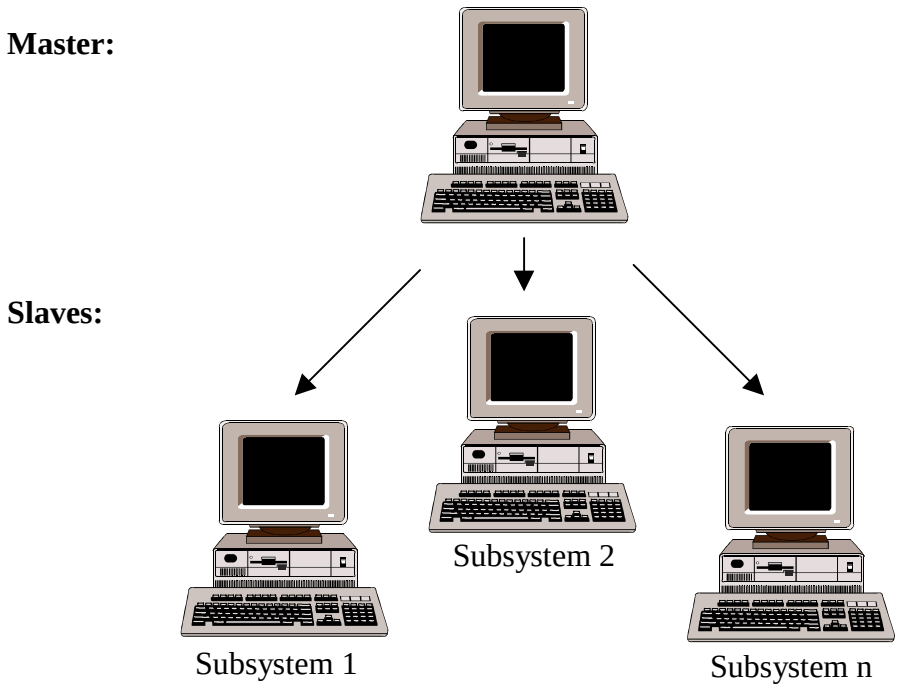


Figure 3.4: Parallel analysis convergence via PVM

Unfortunately, the unaltered PVM results did not live up to expectations. Solution times for parallel convergence of test systems were larger than those corresponding to single machine convergence. Moreover, solution times per iteration were larger, when a *larger* number of slave computers were used on the virtual machine. It is hence probable that the time required to pass information from the master machine to the slave machines is what dominated the convergence time for the PVM scenarios.

To reduce the impact of message passing, "sleep times" were introduced into the convergence procedure. On both the single computer and distributed computing scenarios, each subsystem analysis was performed and then followed by a user-specified period of sleep. At 0.1 seconds of sleep time, the advantages of distributed computing are first detected for all test systems examined, which ranged in size from 1000 to 5000 behavior variables. With 0.5 seconds of sleep time, the advantages of using a larger number of slave machines for distributed computing start to become evident.

Subsequent to the convergence process of the system analysis equations is the sensitivity analysis. In the present research, this is an analytical procedure, and is discussed in the next section.

Sensitivity Analysis

Once the system has converged, CASCADE utilizes the GSE approach, equation [2.6], to attain the total derivative matrix of the system that has been created, around the converged solution point. The left and right hand side partial derivative matrices are computed at the subsystem level. This is made possible by the CASCADE-generated test systems as the sensitivity of each output equation of the system can be analytically computed, with respect to both a) every input behavior variable and b) every subsystem

design variable. These matrices are then normalized, and an LU-Decomposition [65] is used to attain the normalized total derivative matrix (the GSM). The total derivatives are recovered by "un-normalizing" each element of the GSM.

The potential utilization of this GSM matrix depends on the specific research needs of the user. This matrix could be of use to the design manager from several vantage points. For instance, the matrix could be used to assess the coupling relations that are weak relative to the others. It is possible that these weak couplings could be eliminated from the system analysis, or at least suspended for part of it, allowing for computational savings. These derivatives could also be used for a formal, gradient-based system optimization procedure, such as that which was outlined in Figure 2.7.

CASCADE - Optimization Feature

The system analysis and sensitivity analysis features represented a strong groundwork for the initial inception of CASCADE. However, a final necessary component for a full-fledged MDO system simulator is the capability to simulate a multidisciplinary optimization procedure. An optimization feature would thus allow for the simulation of the entire MDO design cycle depicted in Figure 2.7. To utilize CASCADE for conducting meaningful large-scale simulation-based multidisciplinary research, this is a feature that must be present, and has since been added to the simulator [39] for this very reason.

The present version of CASCADE allows the user to generate an optimization problem whose variables are the coupled variables of the converged system analysis. These variables include design variables, which were initialized and held constant for the iterative system analysis, and behavior variables, which were initialized, iteratively

updated, and eventually converged. During the optimization portion of a multidisciplinary design cycle, the behavior variables are treated as constants, and the design variables are altered in search of the global optimum solution.

When designing a simulated set of optimization functions from scratch, certain provisions must be made to guarantee that an optimum solution will exist for the problem that is generated. A large concern would be the possibility of generating an over-constrained design problem. A simple (albeit somewhat unrealistic) solution to this problem is to allow CASCADE to only generate inequality constraint functions; no equality constraint functions are generated. In an equality constrained problem, the number of equality constraints "l" must be less than or equal to the number of design variables "n". This ensures that the number of unknown quantities is less than or equal to the number of equations present. By eliminating the possibility of equality constraints in the system simulation, a degree of realism is lost in favor of a high degree of robustness gained.

As for the semantic nature of the constraints present, the computer simulator development of Padula et al. [61,62] demonstrated that each constraint function must be one of four forms. These forms are: (a) strictly decreasing (exponent: -1), (b) strictly increasing (exponent: 1), (c) asymptotically increasing (exponent: 1/2), and (d) quadratic (exponent: 2). It may be argued that these four polynomial types represent a large subset of optimization functions encountered in engineering design applications. CASCADE implements these heuristics by enforcing that the constraint term exponents be one of these four types. Inequality constraints are posed at the subspace level within CASCADE; no system level design constraints are posed.

Side constraints of [-9999, 9999] for each optimization variable are also prescribed. The optimization functions are implicitly generated to behave within these bounds. Further, the optimization problem is generated such that the converged design point attained during the system analysis is a feasible point in the corresponding optimization problem.

The objective function that is generated can be system-level in nature, or the user can generate a distinct objective function for each subspace. The user can control the state of the coupling between the subspaces for the latter objective function type. The user can enforce that all design variables allocated to subspace i only appear in the optimization functions belonging to subspace i. Alternatively, the user can simulate the more realistic scenario: that design variables are required inputs to functions of numerous subspaces. Such would be useful for CSSO-based research.

Example System - Optimization Problem Representation

Recall that the CASCADE-generated optimization problem is a function of the same design and behavior variables that are found in the analysis equations. Hence, pertaining to the example analysis equations represented by Figure 3.3 and equation [3.6], a representative optimization problem might appear as follows:

$$\begin{aligned} \text{Minimize} \quad & F = 0.05X_V - 4.4\text{e-}5X_W^2 + 5.8\text{e-}10X_Y^3 - 0.88X_Z + \\ & \qquad 2.3\text{e-}9V^3 - 5.3\text{e-}5W^2 - 0.46Y + 7.7\text{e-}9Z^3 \\ \text{subject to} \quad & g_{V1} = -0.48X_V - 3.3\text{e-}5Z^2 \le 0 \\ & g_{Y1} = 6.6\text{e-}10X_Y^3 + 5.9\text{e-}10V^3 - 0.23W + 0.18Z \le 0 \\ & g_{Z1} = -2.6\text{e-}4X_Z^2 + 6.0\text{e-}5V^2 - 9.8\text{e-}9W^3 - 0.07Y^3 \le 0 \\ & -9999 \le X_V, X_W, X_Y, X_Z \le 9999 \end{aligned} \qquad \textbf{[3.7]}$$

Here, subspaces V, Y, and Z have 1 inequality constraint each; subspace W is unconstrained. Note that the objective function is system level, and is a function of every design and behavior variable in the system. Further note that the constraint functions are functions of the same variables as their corresponding analysis equations were. For example, the constraint in subspace V is a function of both design variable X_V, and behavior variable Z, as governed by the boolean matrix in equation [3.5]. The simulator was coded this way initially simply for the sake of convenience. Note however, that the degree of realism in this initial arrangement is lacking. As mentioned in Chapter 2, the subsystem and subspace variables typically do not identically coincide.

Lastly, recognize that in the problem formulation corresponding to the standard, MDF design cycle (Figure 2.7), only the design variables **X** are treated as changeable parameters in the optimization procedure. The side constraints reflect this fact accordingly.

This chapter concludes on a discussion of some of the major features that have been added to CASCADE for increased realism and usability.

CASCADE - Expanded Features and Robustness

The CASCADE simulator is in a continual process of design and development. in order to generate test systems of sufficient realism and robustness, numerous enhancement to the simulator were required. Some of these additions and changes have already been incorporated into the discussion. Those that have not are presented now for completeness sake.

One of the first modifications to CASCADE was the addition of “double coupling” or *interaction* terms to the analysis and optimization functions. Numerous

researchers have suggested that the realism of CASCADE could be increased with the presence of such terms. This is because the semantic of a given behavior variable will likely not only be the summation of effects of its individual input variables, but also the effect of dynamics *between* its input variables. To allow for interaction terms, the generalized equation model of equation [3.1] is supplemented with the following terms:

$$y_k = \cdots + \sum_{i=1}^{q} \sum_{j=1}^{r} c_{ij} (\Psi_i^{e_i} \Psi_j^{f_j}) \qquad k = 1, p$$

[3.8]

Here, c_{ij} is the coefficient of the interaction term, e_i and f_j are the exponents of the individual couplings, and generic symbol ψ is used to represent the couplings in the interaction term. Realize that this couplings could be either a design variable (x) that serves as input to the given behavior variable y_k, or a behavior variable input term (y_s), where $s \neq k$. Lastly, note that q and r correspond to the total number of coupling (design or behavior) variables represented by ψ_i and ψ_j, respectively.

The second major modification to CASCADE involves the expansion of the boolean capabilities for the analysis and the optimization. The initial version of CASCADE only allowed the user to specify couplings between the subsystems, in the form of a "Subsystem-Subsystem" (SS) boolean. This does not take into account the number of outputs (behavior variables) per subsystem. For some researchers, this may be sufficient. However, some researchers might need to specify the coupling nature of the system analysis to a more specified level of decomposition. Hence, the capability of creating a "Behavior variable-Behavior variable" (BV) boolean has been instituted. Consider the system shown in Figure 3.4.

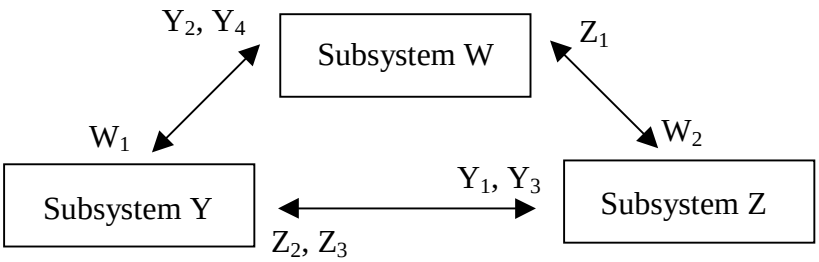


Figure 3.5: Example system (booleans)

The decomposed system has 3 subsystems, W, Y, and Z. Further, subsystem W has 2 behavior variables W_1 and W_2, subsystem Y has 4 behavior variables Y_1, Y_2, Y_3, and Y_4, and subsystem Z has 3 behavior variables Z_1, Z_2, and Z_3. The corresponding SS and BV booleans appear in equations [3.9] and [3.10], respectively. At the subsystem level, the system is fully coupled. At the behavior variable level, the system is NOT fully coupled.

	W	**Y**	**Z**
W	0	1	1
Y	1	0	1
Z	1	1	0

[3.9]

	$\mathbf{W_1}$	$\mathbf{W_2}$	$\mathbf{Y_1}$	$\mathbf{Y_2}$	$\mathbf{Y_3}$	$\mathbf{Y_4}$	$\mathbf{Z_1}$	$\mathbf{Z_2}$	$\mathbf{Z_3}$
$\mathbf{W_1}$	0	0	1	1	1	1	0	0	0
$\mathbf{W_2}$	0	0	0	0	0	0	1	1	1
$\mathbf{Y_1}$	0	0	0	0	0	0	1	1	1
$\mathbf{Y_2}$	1	1	0	0	0	0	0	0	0
$\mathbf{Y_3}$	0	0	0	0	0	0	1	1	1
$\mathbf{Y_4}$	1	1	0	0	0	0	0	0	0
$\mathbf{Z_1}$	1	1	0	0	0	0	0	0	0
$\mathbf{Z_2}$	0	0	1	1	1	1	0	0	0
$\mathbf{Z_3}$	0	0	1	1	1	1	0	0	0

[3.10]

Another addition to the simulator is the feature that allows the user to create a separate boolean matrix pertaining to the optimization functions. This is a necessary addition, as the subsystem (analysis) and subspace (optimization) decompositions of an engineering system will rarely, if ever, coincide identically. The user is thus given the option to create a separate optimization boolean whose coupling structure does not coincide with the analysis, or for simplicity sake, can still utilize the analysis boolean for the optimization as well. Pertaining to Figure 3.4, assume that CASCADE is to generate a subsystem-level objective function for each of the 3 subsystems, and that each subsystem has 1 inequality constraint. The subspace boolean will have optimization functions listed down the columns of the matrix, with the subspace objective functions (F) listed first, followed by the subspace constraint functions (G). The behavior variable couplings are listed across the rows. For the present example, this matrix would be of dimensionality (9 x 6), and might appear as follows:

	F_W	F_Y	F_Z	G_W	G_Y	G_Z
W_1	0	1	1	0	1	1
W_2	0	1	1	0	1	1
Y_1	1	0	1	1	0	1
Y_2	1	0	1	0	0	1
Y_3	1	0	1	1	0	0
Y_4	1	0	1	1	0	1
Z_1	1	1	0	0	0	0
Z_2	1	1	0	0	0	0
Z_3	1	1	0	1	0	0

[3.11]

The third major addition to the CASCADE simulator involves the stability of the analysis equations generated. Recall that the default equations are constructed in such a way that stability of all variables within an implicit set of bounds is guaranteed. Refer to

equations [3.3] and [3.4]. The trade off for this stability is that the systems that result often converge in a relatively small number of fixed-point iterations, usually less than 10. Certain types of MDO research might require test systems that are of an unstable nature to a certain degree, or require a larger number of iterations to attain convergence, or both. To allow for this, a user-defined parameter called "system volatility" has been introduced into the CASCADE simulator. The user can pre-define the volatility (instability) of the system prior to generating the analysis equations. The allowable range is 1 through 100. A system volatility of 10 is set as the default value, at which a stable test system is guaranteed. Any value below 10 will result in a highly stable system. Values greater than 10 will result in increasingly more unstable and more oscillatory test systems.

CASCADE might not be able to create a system of a specified level of stability on its first try. If this is the case, the structure of the desired system is retained, and the semantics are randomly altered, and the entire process repeats itself. Many "system construction" attempts could be required. The simulator allows the user to specify the desired *minimum* number of iterations required to converge the system of specified volatility. Because the process of identifying this system might be a lengthy one, the user must also specify the *maximum* number of attempts to generate such a system, before giving up.

Finally, recall that the parallelization of the analysis convergence procedure using PVM demonstrated computational savings. However, this only occurred after the CASCADE equation solutions were augmented with sleep times, to simulate longer overall calculation times. Clearly, the raw solution times of the CASCADE-generated equations are not, in most cases, sufficiently long for conducting "cost-based" MDO

research. Hence, a more sophisticated means to account for this disparity would be desirable. For this reason, a cost module for both analysis and optimization function evaluations has been added to the simulator. This module allows the user to assign a generic "cost" quantity to each behavior variable function, and each optimization function present in the system. These costs are then written to a data file. Whenever a function evaluation takes place thereafter, be it during the system analysis or during an optimization, the associated cost can be summed and accumulated.

These costs could be of tremendous use for MDO method comparison studies. This will become evident in the next chapter, which presents a formal discussion of this very topic.

Chapter 4

Multidisciplinary Solution Strategies

The present chapter presents a discussion and comparison of numerous approaches in MDO for posing and subsequently solving multidisciplinary design problems. The decomposition of a large engineering system is typically hybrid-hierarchic in nature, thus necessitating a full iterative convergence of the analysis equations, **each and every MDO cycle**. Hence, the primary motivation behind identifying alternative approaches to the standard MDO solution cycle (illustrated in Figure 2.7) is to reduce this large associated analysis cost. The standard MDF solution strategy is presented first, followed by two of the most popular alternative solution methods. Thereafter, a sample CASCADE system is generated, and the design problem is then posed for each of the three solution strategies. Finally, the chapter concludes with a presentation of results attained through the use of CASCADE for method comparison study.

Multiple-Discipline Feasible (MDF)

Consider the decomposition of a generic aerospace system into a coupled grouping of disciplinary subsystems, as shown in Figure 4.1. As alluded to previously, for a true engineering system, the resultant decomposition grouping is typically hybrid-

hierarchic in nature, as seen in the figure. This inherent lack of hierarchy requires that the system analysis associated with the overall design cycle be initialized to some set of values, and iteratively converged thereafter. Subsequent to attaining a converged analysis solution, a sensitivity analysis is performed. The sensitivity analysis can be a numerical procedure such as finite differencing or the Global Sensitivity Equation (GSE) method. The sensitivity analysis is required for a gradient-based optimization of the overall design. The optimization procedure itself will typically cause certain variables to change (hereafter referred to as *optimization variables*), which then necessitates the re-convergence of the system analysis. Hence, the entire design cycle repeats itself until an overall converged solution is attained. A summary of such hybrid-hierarchic design synthesis is illustrated in Figure 4.2.

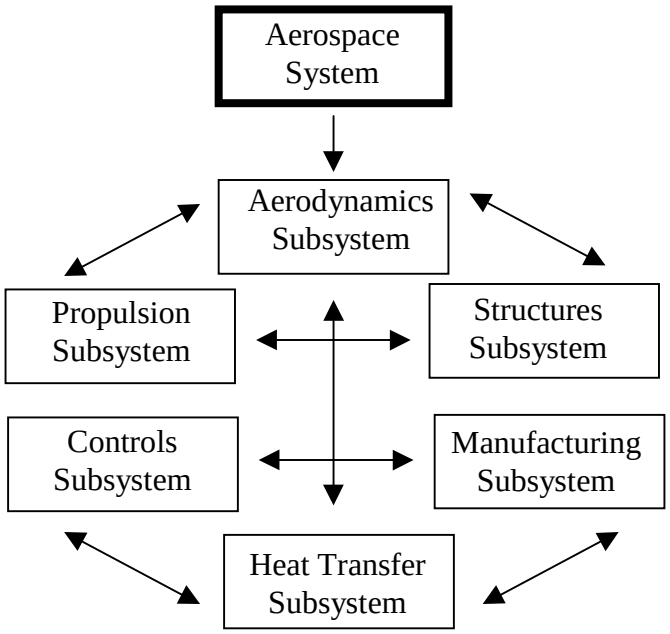


Figure 4.1: Hybrid-hierarchic aerospace system decomposition

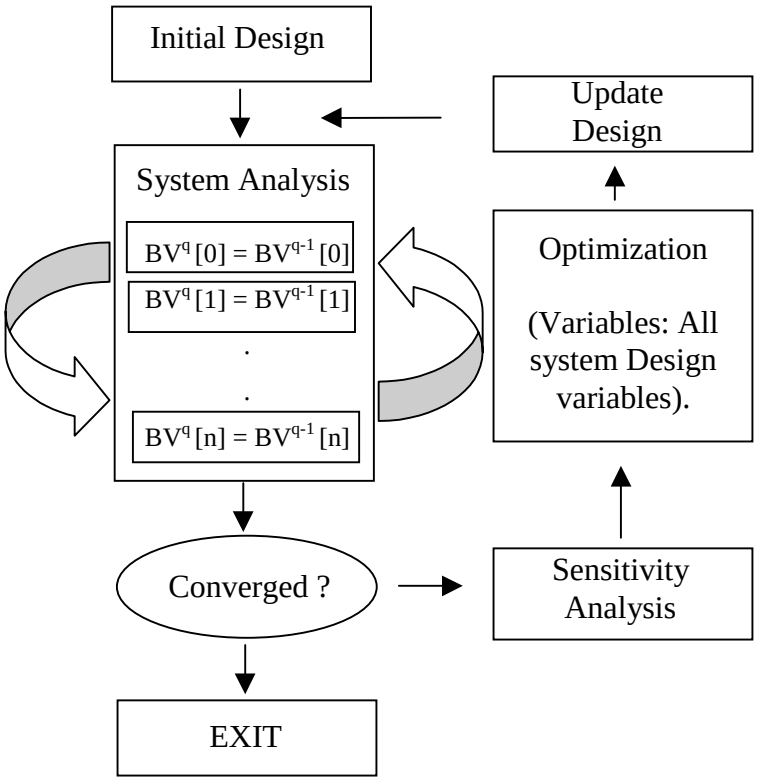


Figure 4.2: The "Multiple-Discipline-Feasible" (MDF) strategy

The design cycle that has just been described has been referred to in literature as the "Multiple-Discipline-Feasible", or MDF approach [14,15,3]. It has been demonstrated more than any other approach on non-hierarchic multidisciplinary examples. The advantages of the MDF approach include its commonality to most MDO researchers, and its optimization problem, which treats only design variables (and **not** behavior variables) as optimization variables. The primary disadvantage of the MDF approach is that it is potentially very time and cost consuming. Complete multidisciplinary feasibility is enforced at each optimization iteration. At each design cycle, a great deal of time may be inefficiently spent while fully re-converging the system analysis portion of a design that is still very far from its eventual, optimal solution.

All-at-Once (AAO)

More recently, researchers have focused on alternate methods for posing and solving the multidisciplinary design problem. An approach has been developed which treats the entire multidisciplinary design cycle seen in Figure 4.2 as a single large optimization problem. This is accomplished by converting the system analysis equations into equality constraints, and by treating both system design variables and subsystem outputs (behavior variables) as optimization variables. Such an approach has been referred to in literature as "All-at-Once" (AAO) [14,15,3].

The primary advantage of AAO is the elimination of the iterative design cycle for attaining an optimal design. This occurs through the outright elimination of costly iterative analysis evaluations. One possible disadvantage of AAO is that a much more complicated optimization problem results. More optimization variables and more equality constraints are present in the AAO optimization problem formulation. These variables and equations stem from the addition of the system analysis equations to the optimization problem statement. A second disadvantage is that disciplinary feasibility is only attained at a relative or at an absolute extremum. Hence, if the optimizer is unsuccessful in attaining the global optimum solution, the likelihood of completing the solution cycle with a feasible design solution is reduced.

A generalized summary of the AAO strategy is seen in Figure 4.3. Notice that the "Residual Evaluator" has replaced the iterative System Analysis seen in Figure 4.2. In the Residual Evaluator, the analysis equality constraints are posed.

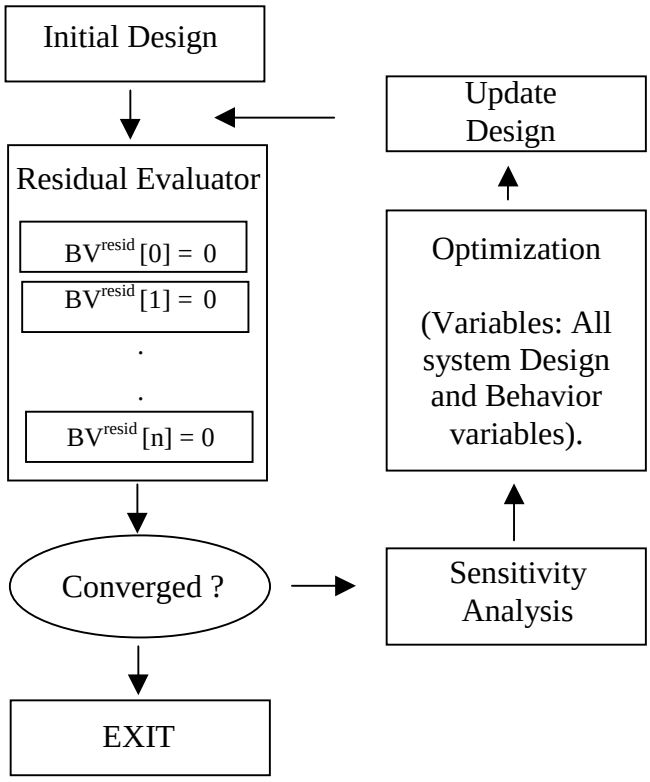


Figure 4.3: The "All-at-Once" (AAO) strategy

Individual-Discipline Feasible (IDF)

A second popular alternative to the classical MDF approach exhibits characteristics which lie between the extremes of MDF and AAO. Recall that MDF requires full disciplinary feasibility at each and every optimization iteration, while AAO only enforces disciplinary feasibility at the final solution (a local or global optimum), if attained. An intermediate approach is called "Individual-Discipline-Feasible" (IDF) [27,14,15,3]. With IDF, each individual discipline (or subsystem) is *independently feasible* at each optimization iteration. The optimizer eventually drives all of the individual disciplines towards multidisciplinary feasibility by controlling the interdisciplinary data. In this formulation, all *coupling variables* (behavior variables that are required inputs to other subsystems) are promoted to being optimization variables.

This takes place by temporarily substituting a replacement "surrogate" variable for each coupling variable in the optimization problem statement. Auxiliary equality constraints are added to the problem formulation to ensure that each and every behavior variable is equal to its corresponding surrogate variable, at optimization convergence. These constraints may be thought of as "equilibrium" constraints. A generalized summary of the IDF strategy is seen in Figure 4.4. Notice that the "Analysis Solver" has replaced the iterative System Analysis seen in Figure 4.2. In the Analysis Solver, two things take place: the single analysis solution (non-iterative) of each and every behavior variable equation, and the formulation of the equilibrium constraints.

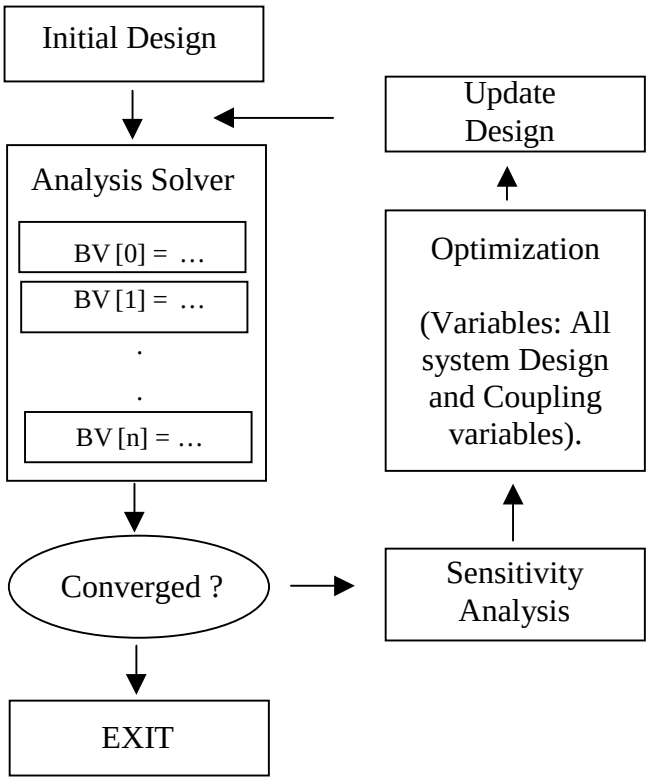


Figure 4.4: The "Individual-Discipline Feasible" (IDF) strategy

It should be noted that the nomenclature used to describe the primary MDO solution strategies in this research effort is not exclusive terminology used in the MDO

community. The next section briefly discusses two of the more recently developed alternate nomenclatures.

Alternate Terminology

Numerous researchers have developed related, but alternate naming schemes to describe the methods. One of the first known researchers to implement the "SAND" terminology and ideology was Haftka [26] for nonlinear structural analysis. In this study, he treats all response variables and structural parameters as design variables in a unified optimization formulation. In so doing, analysis and design are performed simultaneously. Braun et al. [9] conducted a comparison of the MDF and AAO solution strategies for launch-vehicle design. In this study, the MDF strategy is referred to as the "Iterative-Loop Method", and AAO is referred to as the "Sequential Compatibility Constraint Method".

Balling and Sobieski [3] have devised a nomenclature based on the control of the variables at both the system and disciplinary levels. *Simultaneous Analysis and Design* (SAND) implies that the disciplinary design and coupling variables are determined simultaneously by the optimizer. Alternatively, *Nested Analysis and Design* (NAND) implies that the optimizer determines only the disciplinary design variables, and requires determination of the coupling variables each iteration. Hence, MDF has been termed the "NAND-NAND" approach, with the first NAND corresponding to system level, and the second NAND corresponding to the disciplinary level. Similarly, "AAO" has been termed "SAND-SAND", and IDF, the intermediate approach, has been termed "SAND-NAND".

In a recent research effort [2], a completely new taxonomy has been proposed, which is based on the way that the formulation handles the design constraints. These constraints can be one of three possible forms: a) disciplinary analysis constraints, which are equality constraints implicit in disciplinary analyses; b) general nonlinear design constraints, and c) interdisciplinary consistency constraints, introduced to relax interdisciplinary coupling. Note that the latter form of constraints are analogous to the aforementioned "equilibrium constraints". The authors have used this nomenclature to partition the classification of the solution methods as follows: a/b/c. For example, the MDF approach has been termed CDA/OD/CIC, which stands for: Closed Disciplinary Analysis/Open Design constraints/Closed Interdisciplinary Consistency constraints. Similarly, AAO has been termed OA/OD/OIC, which stands for: Open Analysis/Open Design constraints/Open Interdisciplinary Consistency constraints. Finally, the IDF strategy has been termed CDA/OD/OIC, which stands for: Closed Disciplinary Analysis/Open Design constraints/Open Interdisciplinary Consistency constraints.

The present research work recognizes the merits of these alternate nomenclature schemes, but continues to make use of standardized terminology for the remainder of this effort. With a fundamental understanding of each of the three primary solution strategies, the next section presents an example CASCADE test system and a corresponding multidisciplinary problem statement for each.

Integration of CASCADE with the Primary Solution Strategies

A number of previous research efforts involving the comparison of the MDF, IDF, and AAO solution strategies have been extensive in theoretical detail. However, many of these research efforts have limited their implementation of the theory to

"simple" 2-3 subsystem example problems [15,1], often with easily obtainable closed-form solutions [4]. The importance of these past studies cannot be understated. In the present work, it is desired to assess the utility of these solution strategies, side-by-side, over a wide variety of system sizes and subsystem-level coupling densities. Through the use of CASCADE to generate analytical test systems, the desired problem has a known structure, and unknown (randomly generated) semantics. Moreover, the global optimum solution of these test problems is unknown, a priori. It is only after such testing is completed that heuristic rules can be developed which might help to govern the appropriateness of a given solution strategy for a given set of coupled system characteristics.

Figure 4.5 consists of a multidisciplinary system that has been decomposed into three coupled subsystems.

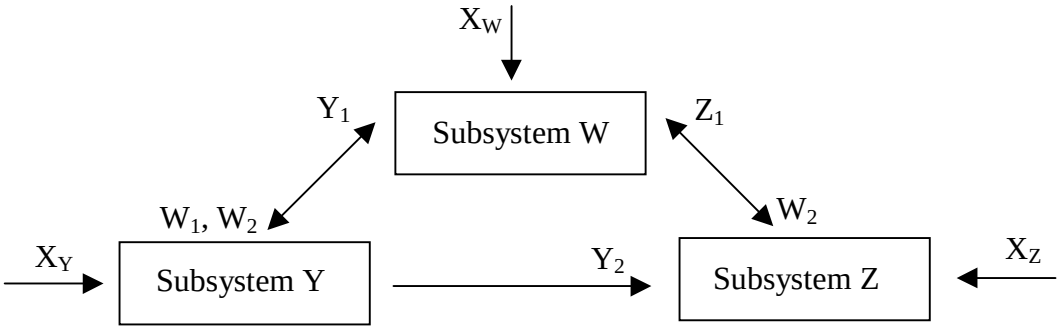


Figure 4.5: Three coupled subsystems

Each subsystem has its own set of independent design variables as input (the "X's"), as well as dependent behavior variables - outputs from other subsystems - also serving as input (the "W's", "Y's", and "Z's"). The CASCADE-generated analysis equations that exhibit the coupling structure illustrated in Figure 4.5 might appear as in equation [4.1]:

$$W_1 = 0.22X_W^1 + 0.05Y_2^3 - 0.46Z_1^2 + 0.73(X_W Y_2)^1$$
$$W_2 = -0.96Y_2^2 + 0.56Z_1^3 - 0.03(X_W Y_2)^2$$
$$W_3 = 0.36X_W^2 + 0.93(Z_1 Y_2)^2$$
$$Y_1 = 0.08X_Y^3 - 0.05W_1^1 + 0.11W_3^1 - 0.09(X_Y W_1)^3$$
$$Y_2 = 0.59W_3^1 + 0.41W_1^2 + 0.99(X_Y W_3)^3$$
$$Z_1 = -0.43X_Z^2 - 0.88W_2^2 + 0.25(W_2 X_Z)^2$$

[4.1]

Note that each behavior variable output equation is a polynomial function of both single-coupling terms and interaction terms. The CASCADE-generated optimization problem is a function of the same design and behavior variables that are found in the analysis equations, and might appear as in Equations [4.2] and [4.3]:

Minimize:

$$F = .04X_W^2 + .96X_Y^3 + .15X_Z - .26W_1^2 + .44W_2 + .57W_3^3 - .07Y_1 + .68Y_2^2 - .02Z_1^3$$

[4.2]

Subject to:

$$g_W = -578.9 + 0.36Y_2^3 + 0.55\ X_W^1 + 0.09(X_W Z_1)^3 \leq 0$$
$$g_Y = -226.7 + 0.26X_Y^3 + 0.51\ W_1^2 + 0.53(X_Y W_3)^1 \leq 0$$
$$g_Z = -1095.1 + 0.33\ Y_2^2 + 0.47(X_Z W_2)^1 \leq 0$$
$$-9999.0 \leq X_W, X_Y, X_Z \leq 9999.0$$

[4.3]

Note again the presence of both single-coupling and interaction terms in the objective and inequality constraint functions. Given this example problem, the problem formulations corresponding to the three solution strategies are presented.

a. MDF

The MDF problem formulation for the present example system is described as follows:

- Optimization variables: X_W, X_Y, and X_Z
- Analysis: Equation [4.1]. Full iterative re-convergence every MDO cycle.
- Optimization: Equations [4.2] and [4.3].

- Comment: The problem is solved as posed in equations [4.1] through [4.3] and in a cyclic manner similar to that seen in Figure 4.2.

b. IDF

The IDF problem formulation is described as follows:

- Optimization variables: X_W, X_Y, X_Z, and surrogate variables X_{W1}, X_{W2}, X_{W3}, X_{Y2}, and X_{Z1}
- Analysis: A single non-iterative "solution" of Equation [4.1] on each cycle, modified as follows. Note the presence of replacement "surrogate" variables on the right hand side of the equations:

$$W_1 = 0.22X_W^{1} + 0.05X_{Y2}^{3} - 0.46X_{Z1}^{2} + 0.73(X_W X_{Y2})^{1}$$
$$W_2 = -0.96X_{Y2}^{2} + 0.56X_{Z1}^{3} - 0.03(X_W X_{Y2})^{2}$$
$$W_3 = 0.36X_W^{2} + 0.93(X_{Z1} X_{Y2})^{2}$$
$$Y_1 = 0.08X_Y^{3} - 0.05X_{W1}^{1} + 0.11X_{W3}^{1} - 0.09(X_Y X_{W1})^{3}$$
$$Y_2 = 0.59X_{W3}^{1} + 0.41X_{W1}^{2} + 0.99(X_Y X_{W3})^{3}$$
$$Z_1 = -0.43X_Z^{2} - 0.88X_{W2}^{2} + 0.25(X_{W2} X_Z)^{2}$$

[4.4]

- Optimization:

Minimize:

$$F = .04X_W^{2} + .96X_Y^{3} + .15X_Z - .26X_{W1}^{2} + .44X_{W2} + .57X_{W3}^{3} - .07X_{Y1} + .68X_{Y2}^{2} - .02X_{Z1}^{3}$$

[4.5]

Subject to:

$$g_W = -578.9 + 0.36X_{Y2}^{3} + 0.55X_W^{1} + 0.09(X_W X_{Z1})^{3} \leq 0$$
$$g_Y = -226.7 + 0.26X_Y^{3} + 0.51X_{W1}^{2} + 0.53(X_Y X_{W3})^{1} \leq 0$$
$$g_Z = -1095.1 + 0.33X_{Y2}^{2} + 0.47(X_Z X_{W2})^{1} \leq 0$$
$$0 = X_{W1} - W_1$$
$$0 = X_{W2} - W_2$$
$$0 = X_{W3} - W_3$$
$$0 = X_{Y2} - Y_2$$
$$0 = X_{Z1} - Z_1$$
$$-9999.0 \leq X_W, X_Y, X_Z, X_{W1}, X_{W2}, X_{W3}, X_{Y2}, X_{Z1} \leq 9999.0$$

[4.6]

- Comment: Note that X_{Y1} is not an optimization variable, since it is not used as input by any subsystem in the analysis equations.

c. AAO

The AAO problem formulation is described as follows:

- Optimization variables: X_W, X_Y, X_Z, W_1, W_2, W_3, Y_1, Y_2, and Z_1
- Analysis: None. ("Analysis" is included in the optimization problem).
- Optimization:

Minimize:

$$F = .04X_W^2 + .96X_Y^3 + .15X_Z - .26W_1^2 + .44W_2 + .57W_3^3 - .07Y_1 + .68Y_2^2 - .02Z_1^3 \quad \textbf{[4.7]}$$

Subject to:

$$\begin{aligned}
&g_W = -578.9 + 0.36Y_2^3 + 0.55\,X_W^1 + 0.09(X_W Z_1)^3 \le 0\\
&g_Y = -226.7 + 0.26X_Y^3 + 0.51\,W_1^2 + 0.53(X_Y W_3)^1 \le 0\\
&g_Z = -1095.1 + 0.15Y_1^1 + 0.33\,Y_2^2 + 0.47(X_Z W_2)^1 \le 0\\
&0 = 0.22X_W^1 + 0.05Y_2^3 - 0.46Z_1^2 + 0.73(X_W Y_2)^1 - W_1\\
&0 = -0.96Y_2^2 + 0.56Z_1^3 - 0.03(X_W Y_2)^2 - W_2\\
&0 = 0.36X_W^2 + 0.93(Z_1 Y_2)^2 - W_3\\
&0 = 0.08X_Y^3 - 0.05W_1^1 + 0.11W_3^1 - 0.09(X_Y W_1)^3 - Y_1\\
&0 = 0.59W_3^1 + 0.41W_1^2 + 0.99(X_Y W_3)^3 - Y_2\\
&0 = -0.43X_Z^2 - 0.88W_2^2 + 0.25(W_2 X_Z)^2 - Z_1\\
&-9999.0 \le X_W, X_Y, X_Z, W_1, W_2, W_3, Y_1, Y_2, Z_1 \le 9999.0
\end{aligned} \quad \textbf{[4.8]}$$

- Comment: All design and behavior variables are controlled by the optimizer.

With a general understanding of the three different means for posing the MDO test systems that are generated by CASCADE, the results of several numerical comparisons are seen in the next section.

Results

a. Preliminary Testing

For first phase testing, CASCADE has been used to generate a total of five simulations of coupled multidisciplinary systems [40]. These five test systems vary in size and coupling complexity. The ANSI-C translated version of Automated Design Synthesis (ADS) [85] has been used as the optimization software for these MDO test systems. The strategy - optimizer combination that has been used for the acquisition of all the results in this section is *Sequential Linear Programming - Method of Feasible Directions*. Internal finite difference methods have been used to attain gradient information within ADS. All trial executions were performed on a *SUN Ultra 1 Creator 3D* workstation, under comparable network conditions. The primary characteristics of the five test systems are summarized in Table 4.1. Recall that the number of *coupling variables* for each solution strategy corresponds to the number of behavior variables that are required as input by at least one other subsystem.

Table 4.1: Preliminary test system summary

Test system number	Number of subsystems	Number of behavior variables	Number of design variables	Number of inequality constraints	Number of coupling variables	Initial objective function
1	3	6	6	6	5	-678.71
2	5	9	11	10	8	-43.91
3	10	20	40	20	12	1622.74
4	15	45	45	90	43	-820.18
5	20	100	40	3	92	197.40

Test system #1 has 3 subsystems (W, Y, and Z), which have 2, 1, and 3 behavior variables per subsystem, 3, 2, and 1 design variables per subsystem, and 1, 3, and 2 inequality constraints per subsystem, respectively. The initial value of the objective

function is -678.71. Figure 4.6 provides a detailed illustration of the coupling structure of the first test system. Figure 4.7 compares the Test system #1 objective function histories for all three solution strategies. Table 4.2 summarizes the "best" results attained (where "best" implies *lowest* objective function) for the first test system, for each of the three solution strategies, after numerous trial executions. All solution strategies achieve approximately the same optimal design point, with the MDF strategy attaining the lowest objective function value of -1711.79.

Table 4.2: Solution summary for test systems #1 - #5

Test System	Solution Strategy	Total OV's	Final OF	Active SC's	Active IC's	Run Time (sec.)
1	MDF	6	-1711.79	3	1	0.486
	IDF	11	-1706.45	3	1	1.916
	AAO	12	-1705.15	3	1	2.029
2	MDF	11	-3150.08	4	1	1.094
	IDF	19	-3144.08	4	1	7.245
	AAO	20	-3124.61	4	1	9.146
3	MDF	40	-11859.7	15	4	22.927
	IDF	52	-11407.5	13	6	27.023
	AAO	60	-11316.8	13	6	64.201
4	MDF	45	-10118.8	13	22	57.854
	IDF	88	-8882.84	11	10	307.943
	AAO	90	-8483.45	11	12	260.995
5	MDF	40	-12012.6	26	2	98.179
	IDF	132	-10253.4	13	2	1892.150
	AAO	140	-10980.8	14	2	3098.899
Notes: OV: Optimization variable; OF: Objective function; SC: Side Constraint; IC: Inequality Constraint						

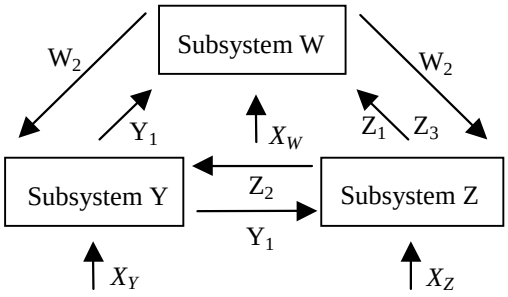


Figure 4.6: Structural schematic of test system #1

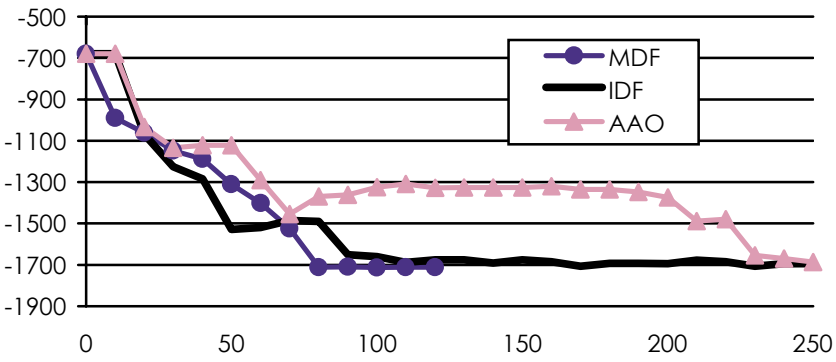


Figure 4.7: Test system #1 - Objective function value vs. evaluation number

Test system #2 has 5 subsystems (U, V, W, Y, and Z), which have 1, 2, 1, 3, and 2 behavior variables per subsystem, 3, 2, 1, 2, and 3 design variables per subsystem, and 4, 0, 1, 2, and 3 inequality constraints per subsystem, respectively. The initial value of the objective function is -43.91. Figure 4.8 provides a detailed illustration of the coupling structure of the test system. Figure 4.9 compares the Test system #2 objective function histories for all three solution strategies. Table 4.2 again summarizes the "best" results attained for the second test system, for each of the solution strategies, after numerous trial executions. Again, all solution strategies achieve approximately the same optimal design point. Here, the MDF strategy attains the lowest objective function value of -3150.08.

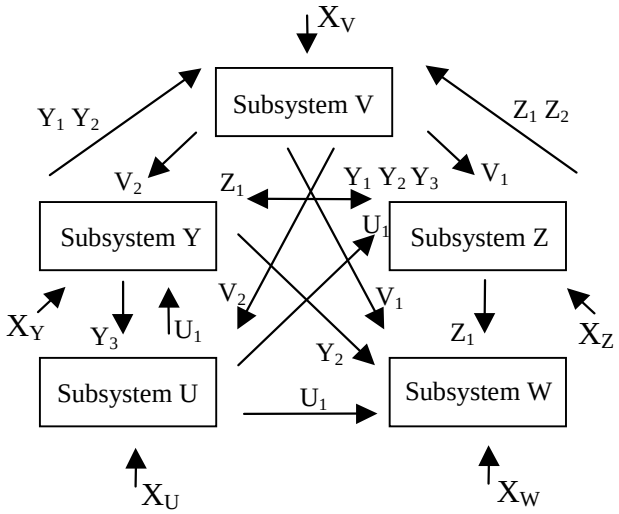


Figure 4.8: Structural schematic of test system #2

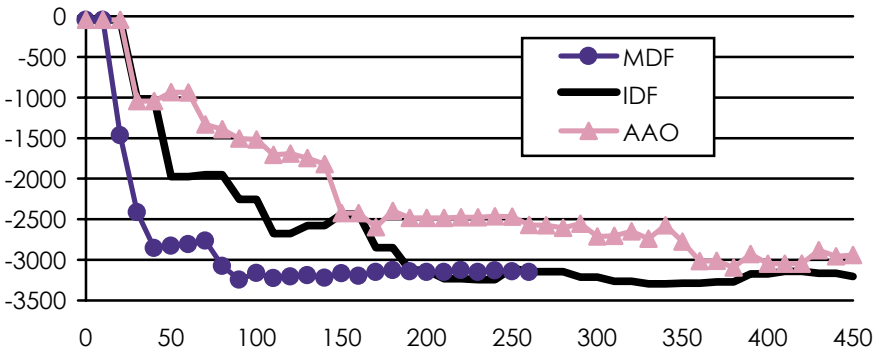


Figure 4.9: Test system #2 - Objective function value vs. evaluation number

Test system #3 has 10 subsystems, a total of 20 behavior variables, 40 design variables, and 20 inequality constraints. For the final three test systems, detailed coupling illustrations and objective function histories are omitted for brevity. The initial value of the objective function is 1622.74. Table 4.2 again summarizes the "best" results attained for the third test system, for each of the three solution strategies, after numerous trial executions. Here again, all solution strategies achieve approximately the same objective function value, but with varying active constraint sets at the final solutions. At the MDF minima, 15 inequality and 4 side constraints are active. At the IDF and AAO minima, 13 inequality and 6 side constraints are active. Numerical constraint "thickness" [85] was set to be the ADS default range of -0.03 to 0.01 for all runs, and hence did not bias any of the three solution approaches. Hence in this case, it is felt that two distinct solution minima have been found. Again, the MDF strategy attains the lowest final objective function value, of -11859.7.

Test system #4 has 15 subsystems, a total of 45 behavior variables, 45 design variables, and 90 inequality constraints. The initial value of the objective function is -820.18. Table 4.2 again summarizes the "best" results attained for the fourth test system,

for each of the three solution strategies, after numerous trial executions. In this case, all three solution strategies do not arrive at "equivalent" solutions. The MDF strategy attains the lowest objective function value of -10118.8, which is approximately 10% lower than the next best solution attained, by the IDF strategy.

Finally, test system #5 has 20 subsystems, a total of 100 behavior variables, 40 design variables, and only 3 inequality constraints. The initial value of the objective function is 197.40. Table 4.2 again summarizes the "best" results attained for the fifth test system, for each of the three solution strategies, after numerous trial executions. Here again, there is a distinct difference between the final solutions reached by each of the three solution strategies. This disparity is likely due to the growing numbers of optimization variables in the IDF and AAO formulations, as the size of the system analysis increases. MDF again achieves the lowest objective function value of -12012.6, considerably lower than the next best solution attained, by the AAO strategy.

The final four figures of this sub-section specifically pertain to the results of test system #5, and provide a visual interpretation of general trends that are evident in a majority of the results that have been presented in this work. Figure 4.10 is a plot of final objective function vs. solution strategy; Figure 4.11 is a plot of total analysis evaluations vs. solution strategy; Figure 4.12 is a plot of total objective function evaluations vs. solution strategy, and Figure 4.13 is a plot of total execution time vs. solution strategy.

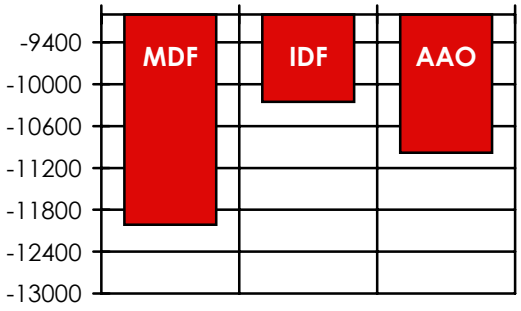


Figure 4.10: Test system #5 - final Objective function

Figure 4.11: Test system #5 - analysis evaluations

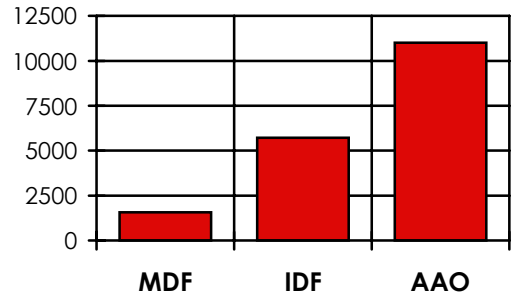


Figure 4.12: Test system #5 - objective function evaluations

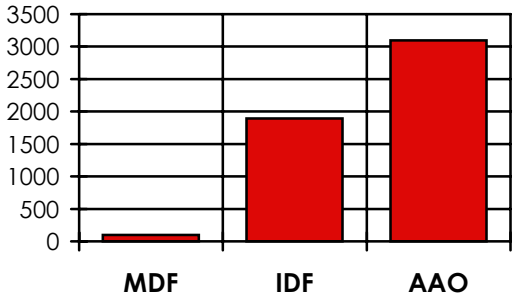


Figure 4.13: Test system #5 - execution time (seconds)

b. Cost-based Testing

The primary theme of this sub-section of results is that of "trade-off". The first sub-section of results focused solely on the ability of each of the solution strategies to achieve design improvement through a decrease in the system-level objective function. However, design improvement is not the only characteristic that must be considered by the design manager. Inevitably, the implementation of each of these solution strategies will have an associated cost. This cost might stem from computational resources expended (i.e. CPU time), from man hours invested in carrying through the solution process of the given system as posed, or from any of a number of other sources. In this research effort, separate simulated and generic cost amounts have been assigned to each analysis evaluation, each objective function evaluation, and each constraint function evaluation. The goal with this study is to arrive at a simulated total cost amount associated with implementing each of the solution strategies [43]. These cost amounts are then compared side by side. The design manager can weigh the gains obtained in the optimized design versus the cost incurred in arriving there, for each of the solution strategies. Three systems were tested for this phase of result acquisition; the characteristics of these systems are summarized in Table 4.3.

Table 4.3: Cost-based test system summary

Test System	# of SS's	# of BV's	# of DV's	# of IC's	# of CV's	Initial OF
1	10	10	10	20	9	-76.37
2	10	20	30	10	18	-577.26
3	10	50	50	5	47	-1436.3
Notes: SS = subsystem; BV = behavior variable; DV = design variable; IC = inequality constraint; CV = coupling variable; OF = objective function						

In this simulation the overall cost in implementing a solution strategy is a function of three distinct components – analysis cost, objective function cost, and constraint function cost. The objective function cost (OFC) is analogous for all three solution strategies; it is the cost of the system objective function, or the sum of the costs of the subsystem-level objective functions, summated every design cycle. The details of the other two components differ slightly for each solution strategy. For the MDF strategy, the analysis cost (AC) is the cost of an analysis evaluation for each behavior variable multiplied by the number of iterations required to converge the system of coupled analysis equations, summated every design cycle. The constraint function cost (CFC) is the sum of the cost of each of the system inequality constraints, summated every design cycle. For the IDF strategy, the analysis cost is the cost of an analysis evaluation for each behavior variable, summated every design cycle. The constraint function cost is the sum of the cost of each of the system inequality constraints, plus the sum of the cost for each of the equilibrium constraints corresponding to each coupling variable, summated every design cycle. For the AAO strategy, there is no explicit analysis evaluation cost. The constraint function cost is the sum of the cost of each of the system inequality constraints, plus the sum of the cost of each analysis equation (where here each analysis equation is

posed as an equality constraint), summated every design cycle. These associated costs are seen in equations [4.9] through [4.12].

$$\text{Total cost (TC)} = \text{OFC} + \text{AC} + \text{CFC}$$

[4.9]

MDF (TC):

$$\text{OFC} = \text{numcyc}^*\left(\sum_{k=1}^{p} \text{of cost}_k\right)$$

$$\text{AC} = \text{numcyc}^*\left(\text{numits}_q * \sum_{i=1}^{m} \text{bv cost}_i\right)$$

$$\text{CFC} = \text{numcyc}^*\left(\sum_{j=1}^{n} \text{c cost}_j\right)$$

[4.10]

IDF (TC):

$$\text{OFC} = \text{numcyc}^*\left(\sum_{k=1}^{p} \text{of cost}_k\right)$$

$$\text{AC} = \text{numcyc}^*\left(\sum_{i=1}^{m} \text{bv cost}_i\right)$$

$$\text{CFC} = \text{numcyc}^*\left(\sum_{j=1}^{n} \text{c cost}_n + \sum_{i=1}^{m} \text{ec cost}_i\right)$$

[4.11]

AAO (TC):

$$\text{OFC} = \text{numcyc}^*\left(\sum_{k=1}^{p} \text{ofcost}_k\right)$$

$$\text{AC} = 0$$

$$\text{CFC} = \text{numcyc}^*\left(\sum_{j=1}^{n} \text{c cost}_j + \sum_{i=1}^{m} \text{ae cost}_i\right)$$

[4.12]

In the above equations, "numcyc" is the total number of MDO cycles required for system convergence, and "$numits_q$" is the number of iterative system analyses required for convergence on MDO cycle q. Moreover, "$ofcost_k$" is the evaluation cost associated with the k^{th} objective function, and "$bvcost_i$" is the evaluation cost associated with the i^{th} system behavior variable. Further, "$ccost_j$" is the evaluation cost associated with the j^{th} system inequality constraint, and "$eccost_i$" is the evaluation cost associated with the i^{th} system behavior variable (posed as an equilibrium constraint). Finally, "$aecost_i$" is the evaluation cost associated with the i^{th} system behavior variable (posed as an analysis equality constraint). Here, "p" is the number of objective function associated with the system design, "m" is the number of subsystem behavior variables, and "n" is the total number of inequality constraints in the system.

Three "cost scenarios" are implemented in the presentation of the results. These scenarios are summarized in Table 4.4. Cost scenario #1 is such that the cost of an analysis evaluation is approximately equivalent to the cost of both an objective function evaluation and an inequality constraint function evaluation. Here, the cost of all three function evaluations are randomly chosen to be between 0 and 100 units. This is clearly not a realistic cost scenario. The cost of an analysis evaluation (which, in a real system might stem from the result of a costly FEM matrix inversion, for example) could be far greater than the evaluation cost of the optimization functions, linear or non-linear. In response to this line of thinking are more realistic cost scenarios, #2 and #3. Cost scenario #2 represents a "semi-costly" analysis, where the costs of analysis evaluations are randomly chosen to be between 0 and 500 units, and objective and inequality constraint function costs are once again randomly chosen to be between 0 and 100 units.

Hence, on average, the analysis/optimization evaluation cost ratio is equal to 5.0 for cost scenario #2. Cost scenario #3 represents a "costly" analysis, where the costs of analysis evaluations are randomly chosen to be between 500 and 1000 units, and objective and inequality constraint function costs are again randomly chosen to be between 0 and 100 units. Hence, on average, the analysis/optimization evaluation cost ratio is equal to 15.0 for cost scenario #3.

Table 4.4: Cost scenarios

Cost scenario	Analysis cost range	Optimization cost range	Analysis/Optimization cost ratio
1	(0,100)	(0,100)	1.0
2	(0,500)	(0,100)	5.0
3	(500,1000)	(0,100)	15.0

Relative to all other cost quantities, the evaluation cost of the equilibrium constraints (in the IDF solution strategy) could be considered negligible. However, in this simulation, the cost of an equilibrium constraint function evaluation is set to be equal to 10% of the evaluation cost of its associated analysis equation.

Table 4.5 presents a comparison of the three solution strategies for all three test systems and cost scenarios. Consider first test system #1, cost scenario #1. Listed for each solution strategy are five quantities: objective function improvement (OFI), analysis evaluation cost (AC), objective function evaluation cost (OFC), constraint function evaluation cost (CC), and total evaluation cost (TC). Figure 4.14 presents a plot of this very same data. Note however, that each of these five quantities are plotted normalized, and are normalized separately. (i.e. the three objective function improvements are normalized against each other, the three analysis costs are normalized against each other, the three objective function costs are normalized against each other, etc.) The general

trends of this first system are as follows: MDF achieves the greatest objective function improvement, but has a huge analysis cost and by far the highest overall cost. AAO attains substantial improvement in the objective function, has no iterative analysis cost whatsoever, but has a large constraint cost, because the analysis equations are now posed

Table 4.5: Cost scenario results

TS	CS	Strategy	OFI	Analysis cost	Objective Function cost	Constraint cost	Total cost
1	1	MDF	2337.	2.16E6	12247	3.40E5	2.51E6
		IDF	1831.	1.69E5	8917	2.61E5	4.40E5
		AAO	2324.	0	16354	7.64E5	7.81E5
	2	MDF	2337.	1.08E7	12247	3.40E5	1.12E7
		IDF	1831.	8.54E5	8917	3.21E5	1.18E6
		AAO	2324.	0	16354	2.01E6	2.03E6
	3	MDF	2337.	2.62E7	12247	3.40E5	2.65E7
		IDF	1831.	2.06E6	8917	4.29E5	2.50E6
		AAO	2324.	0	16354	4.22E6	4.23E6
2	1	MDF	7276.	7.20E6	37710	4.10E5	7.65E6
		IDF	3349.	7.66E5	30915	3.98E5	1.19E6
		AAO	3329.	0	27585	9.83E5	1.01E6
	2	MDF	7276.	3.62E7	37710	4.10E5	3.67E7
		IDF	3349.	3.85E6	30915	6.70E5	4.55E6
		AAO	3329.	0	27585	3.73E6	3.76E6
	3	MDF	7276.	1.00E8	37710	4.10E5	1.01E8
		IDF	3349.	1.07E7	30915	1.28E6	1.20E7
		AAO	3329.	0	27585	9.86E6	9.88E6
3	1	MDF	14581.	2.45E7	3828	2.00E5	2.47E7
		IDF	3302.	2.19E6	2649	3.31E5	2.52E6
		AAO	3106.	0	6063	5.33E6	5.33E6
	2	MDF	14581.	1.24E8	3828	2.00E5	1.24E8
		IDF	3302.	1.10E7	2649	1.18E6	1.22E7
		AAO	3106.	0	6063	2.56E7	2.56E7
	3	MDF	14581.	3.71E8	3828	2.00E5	3.71E8
		IDF	3302.	3.31E7	2649	3.25E6	3.64E7
		AAO	3106.	0	6063	7.61E7	7.61E7
Notes: TS = test system; CS = cost scenario; OFI = objective function improvement							

as equality constraints. In this case, AAO has the largest associated objective function cost. IDF attains substantial improvement (albeit the lowest of the three strategies) at a

low analysis cost, and at the lowest objective function, constraint, and total costs, respectively.

Table 4.5 and Figures 4.15 and 4.16 present analogous results for the more realistic cost scenarios #2 and #3, respectively, for test system #1. In comparing Figures 4.16 and 4.15 to Figure 4.14, the following observations are made. The objective function improvements are the same, and the analysis and objective function costs are higher, but proportionately so. In addition, the constraint cost for IDF grows to be larger than that for MDF, and the proportion by which the total cost for MDF is larger than that for AAO and IDF becomes larger. Table 4.5 presents analogous results for the 2nd and 3rd test systems, respectively.

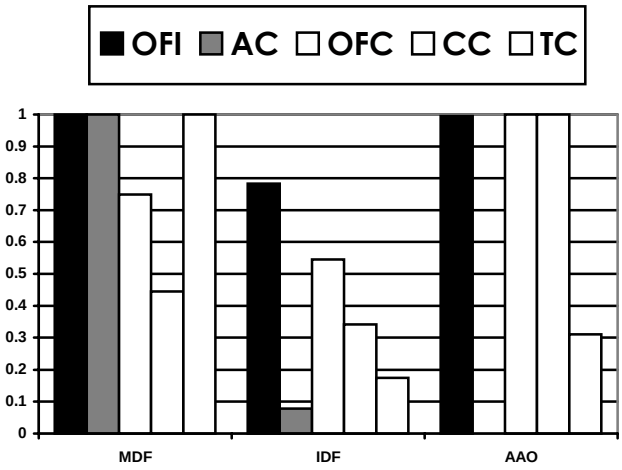


Figure 4.14: Test system #1, cost scenario #1

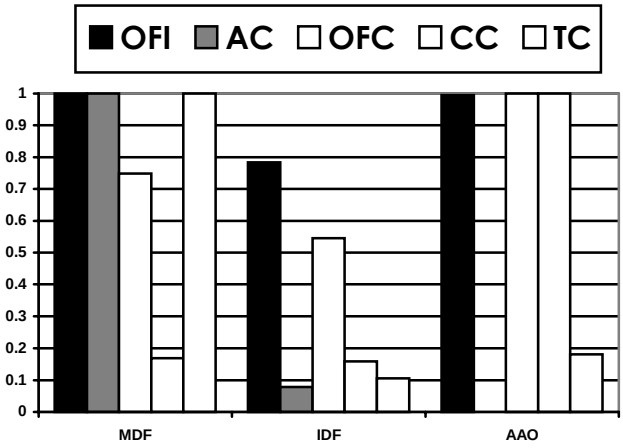


Figure 4.15: Test system #1, cost scenario #2

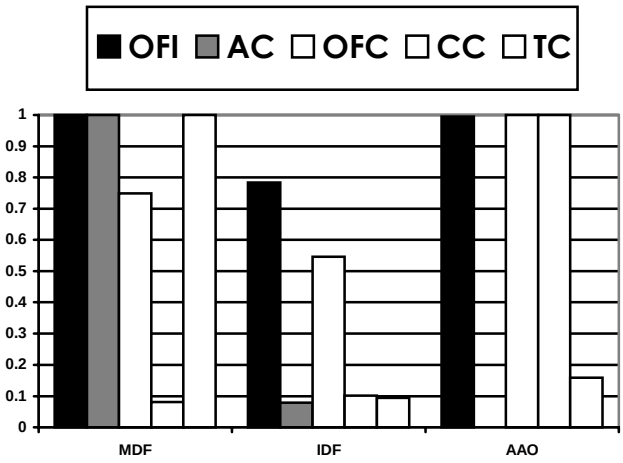


Figure 4.16: Test system #1, cost scenario #3

Figures 4.17 and 4.18 are normalized plots of the cost scenario #3 data for the 2nd and 3rd test systems, respectively. The second test system is typified by the following: the objective function improvement of MDF doubles that of both IDF and AAO, and a huge MDF analysis cost; ten times larger than that of IDF. In addition, the objective function costs were comparable for all strategies, with that of MDF being the largest, followed by IDF and then by AAO. The constraint cost is largest for AAO, followed by

IDF for the second and third cost scenarios (as was the case with the first test system). Finally, MDF has the highest overall cost by an increasing proportion (as cost is increased from cost scenario #1 to cost scenarios #2 and 3), followed by IDF, and followed by AAO, which is the least costly solution strategy for the second test system.

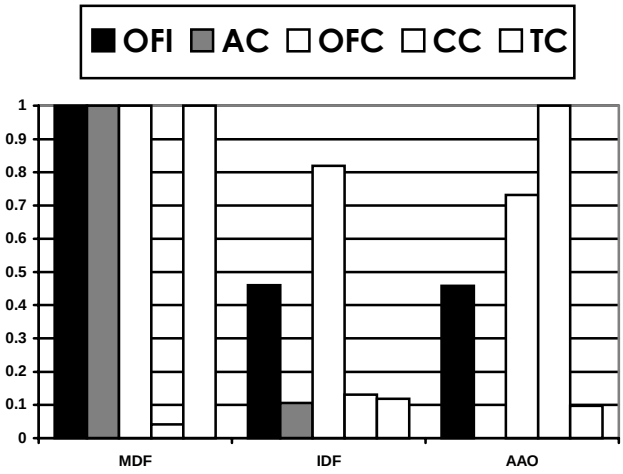


Figure 4.17: Test system #2, cost scenario #3

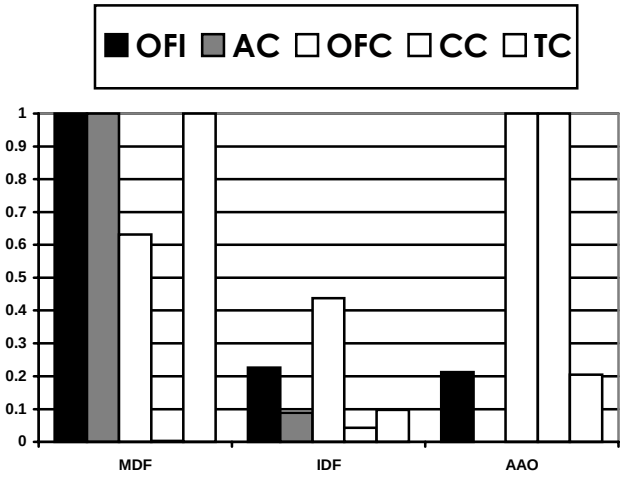


Figure 4.18: Test system #3, cost scenario #3

The third test system is typified by the following: the objective function improvement of MDF that nearly quintuples that of both IDF and AAO, and a huge MDF

analysis cost; ten times larger than that of IDF. In addition, the strategies exhibited comparable (and comparatively miniscule) objective function costs, with that of MDF being the largest, followed by AAO and then by IDF, and the constraint cost is largest for AAO, followed by IDF for all three cost scenarios. Finally, MDF has the highest overall cost by an increasing proportion (as cost is increased from cost scenario #1 to cost scenarios #2 and 3), followed by AAO, and followed by IDF, which is the least costly solution strategy for the third test system.

The next section will take a closer look at both sub-sections of results, and will consequently make some heuristic observations regarding the utility of each of the solution strategies.

Discussion of Results

a. General Observations

The first major point to be made regarding the results is that by having multiple solution strategies to pose and solve the same coupled multidisciplinary design problem, a design manager will arrive at three different problem statements, each with its own "problem dynamic". The solution path for each problem will likely be totally unique, and highly dependent on the initial design point. Hence, having three separate means for posing and solving an MDO problem can only be advantageous to a design manager. This was clearly found to be the case in the present work. Several trial executions would typically be performed on a given test problem using a given solution strategy. Often, the solution that was initially attained appeared to be the "global" optimum solution. Thereafter, numerous trial executions of the same test problem with a different solution strategy would often result in an improved solution, hence revealing the initial "global"

optimum solution as being merely an improved local optimum solution. This type of approach is similar in spirit to attempting the MDO procedure from a variety of different starting design points.

The "problem dynamic" that is mentioned in the previous paragraph results from the nature of each solution strategy; the means with which the multidisciplinary design problem is posed for a given strategy. Problems that contain a large system analysis portion (i.e. problems which have a large number of behavior variables) will result in a large iterative system analysis for the MDF strategy. The same scenario will result in no system analysis at all for AAO, but will instead result in a large number of non-linear inequality constraints in the optimization. This same scenario results in an intermediate state for IDF - an equilibrium equality constraint and a single analysis evaluation are required for each behavior variable that promotes coupling.

A large factor that is affected by this topic of "problem dynamic" is the use of allowable search move limits [7] within the optimizer. CASCADE has been known to generate treacherous design spaces that are cluttered with a multitude of local minima, any of which may contain the initial design point. Often times, very large move limits are required to retreat such a scenario. Because MDF (and to a lesser degree, IDF) has a distinct analysis step, large move limits must be handled with caution. If a design manager allows the design variables to change by a large degree on a given optimization iteration, the ensuing system analysis may see large changes in the behavior variable magnitudes. This may cause the initial design on the subsequent optimization cycle to be infeasible, from which the optimizer might not be able to recover, depending on the

degree of infeasibility. The trial executions of the test systems during both testing solidified these statements.

b. Preliminary Testing

A major area of discussion involves the "goodness" of the results attained. The primary figure of merit for the test results is the final value of the objective function. Figures 4.7 and 4.9 plot the objective function histories of the first and second test systems, respectively. In both of these test systems, the MDF strategy, which has the fewest optimization variables, attains the lowest final objective function, and reaches its final solution the most quickly. The AAO strategy, which has the largest number of optimization variables, tends to lag towards its final solution much more slowly. Not surprisingly, the IDF strategy tends to follow a path that is intermediate to the extremes of the MDF and AAO strategies. In fact, the MDF strategy attains the lowest final objective function value for test systems #3, 4, and 5 as well, as seen in Table 4.2. In all but the fifth test system (where AAO slightly outperforms IDF), the IDF strategy attains the next best final solution, followed closely by the final solution of the AAO strategy.

Seemingly, for the types of problems generated by CASCADE (highly non-linear and non-convex), the solution strategy with the smaller number of optimization variables tends to attain the greatest improvement in the objective function. This is likely because the effectiveness of the optimizer tends to decline as its responsibility increases. With a greater number of optimization variables, there are more gradients that have to be calculated and considered, which creates more sources for error during a gradient-based optimization procedure. This will likely lead the optimization search into an inescapable

local minima, especially in a highly nonlinear optimization problem where the starting design point is unfavorable.

Total execution times for the five test systems tended to follow the same trend - MDF attained its solution the fastest, followed by IDF and AAO. (Refer to Figure 4.13). For the MDF solution strategy (FULL iterative system analysis), total execution times for the five test problems were approximately 0.5, 1.1, 22.9, 57.9, and 98.2 seconds, respectively, where the corresponding number of optimization variables for each test problem is 6, 11, 40, 45, and 40, respectively. For the AAO solution strategy (NO system analysis), total execution times were approximately 2.0, 9.1, 64.2, 261.0, and 3098.9 seconds for the five test systems, corresponding to 12, 20, 60, 90, and 140 optimization variables, respectively. One must keep in mind that these execution times are clock times required for multidisciplinary design *simulations*. To attain an understanding of the total time required to attain a solution for a real-life system (of the same coupling structure that is being simulated by CASCADE), one would have to factor in the relative times required for analysis and objective function calculations for the real-life system itself. Analysis calculations are likely to be the largest factor in terms of total execution time, by a considerable margin, which is not being reflected in the CASCADE test problem simulations. These issues were addressed earlier, and will be further discussed in the next sub-section.

c. Cost-based Testing

Clearly, for the types of systems generated by CASCADE, the MDF solution strategy has shown itself to be the most reliable, regardless of system size or system volatility, for attaining the greatest design improvement. However, the cost-based results

demonstrate that the cost incurred in attaining this improvement is tremendous. Analysis cost is typically ten times (or greater) larger for MDF than for IDF. In general, the factor by which analysis cost for MDF exceeds that for IDF is a function of the number iterations that are required to converge the system of coupled nonlinear analysis equations each design cycle. To a lesser degree, this factor is also a function of the percentage of the system behavior variables that are not coupling variables. When it comes to analysis cost, AAO is undoubtedly the most attractive solution strategy – there is absolutely no *explicit* analysis cost. As the average cost of an analysis evaluation increases between the cost scenarios, the total analysis cost increases for MDF and IDF, but does so in proportion.

Objective function cost tends to be comparable for all three solution strategies, and small when compared to the other costs incurred during solution strategy implementation. Being that AAO has the largest associated optimization problem, it tends to incur the greatest objective function cost. This is not always the case; for test system #2, AAO actually incurs the smallest objective function cost. The objective function cost is clearly a function of the dynamic of the problem being solved - the amount of design improvement that is possible by the particular solution strategy being implemented. This is clearly a function of the initial starting design point. As the average cost of an objective function evaluation increases between the cost scenarios, the total objective function cost increases for all three solution strategies, but does so in proportion.

Constraint function cost exhibits the most interesting and noteworthy behavior when comparing the solution strategies and cost scenarios. For MDF, the constraint cost

is solely a function of inequality constraint evaluation cost, which remains constant for all three cost scenarios. For IDF, the constraint cost is a function of both inequality constraint evaluation cost (constant) and equilibrium constraint cost for each coupling variable. Recall that the latter cost has been approximated to be equal to 10% of the analysis cost for each analysis equation (behavior variable) that corresponds to each coupling variable – this quantity changes between cost scenarios. Similarly, for AAO, the constraint cost is a function of both inequality constraint evaluation cost (constant) and the cost of each analysis equation posed as an equality constraint. Again, this latter quantity changes between cost scenarios. Hence, constraint cost is the only of the three cost quantities that changes disproportionately between the cost scenarios.

For example, for test system #1, cost scenario #1, AAO has the largest constraint cost, followed by that of MDF, which is about half as large, followed by IDF, which is about one-third as large as that of AAO. For cost scenario #2, where inequality constraint cost remains constant but analysis cost (on average) increases five-fold over cost scenario #1 (refer to Table 4.4), the AAO constraint cost becomes almost three times larger (than its cost scenario #1 value). At the same time, MDF constraint cost remains constant, and IDF constraint cost nearly matches that of MDF. Finally, for cost scenario #3, the AAO constraint cost doubles from that of its cost scenario #2 value, MDF constraint cost again remains constant, and IDF constraint cost overtakes that of MDF by approximately 25%. For this cost scenario, the AAO constraint cost is almost ten times larger than that of IDF, which has the second highest total constraint cost.

Total cost is a function of all three simulated costs that have been discussed thus far. Total cost was found to be highest for all test systems and all cost scenarios for the

MDF strategy, and usually by a substantial margin. This is due almost entirely to the enormous iterative analysis cost that is incurred every design cycle. Analysis cost is also found in the IDF and AAO strategies. In the AAO strategy, the "analysis cost" is actually a vast portion of the "constraint cost", as the analysis equations are posed as equality constraints in this formulation. However, recall that analysis cost is non-iterative in nature in IDF and AAO, resulting in a lower summated final cost. Constraint cost was the only cost quantity that increased disproportionately between the cost scenarios. As explained previously, this is because the constraint cost for the IDF and AAO strategies is a function of both the constant inequality constraint evaluation cost, and the non-constant analysis evaluation cost, which appears by virtue of the equilibrium constraints and the analysis-based equality constraints, respectively. Objective function cost was found to have a miniscule effect on the total cost for all test problems and cost scenarios.

For the first and third test systems, total cost was higher for AAO than it was for IDF. This initially came as a mild surprise, but upon closer observation, began to make sense. For the first test system, MDF and AAO performed comparably in terms of objective function improvement. IDF showed improvement, but substantially less than that of the other two strategies. Hence, IDF likely converged upon its inferior solution relatively quickly, with a great deal fewer function evaluations (both analysis and optimization) expended. For the third test system, IDF slightly outperformed AAO in terms of objective function improvement. Here again, it is suspected that IDF reached its final solution (which appears to be a troublesome local optimum) rather quickly, and with fewer total evaluations, than did AAO. Here, AAO achieved roughly the same final solution, but its formulation has more optimization variables and a more complicated

design space than does IDF. In both cases outlined above, IDF would hence have a lower associated total cost than would AAO.

A final observation to be made involves objective function cost, which appears to be a good gauge for determining the solution strategy that will incur the lowest total cost. For the first and third test systems, the objective function cost (and hence the total number of objective function evaluations) for the AAO strategy is almost twice as large as that for the IDF strategy. In both test systems, the total cost for the AAO strategy approximately doubled that of the IDF strategy. For the second test system, the objective function cost for the IDF strategy was slightly larger than that for the AAO strategy. For this test system, the total cost of the IDF strategy was slightly larger than that for the AAO strategy. As previously explained, MDF was found to have the largest total cost, often by a substantial margin, regardless of the objective function cost.

This chapter has focused a great deal on the overall MDO cycle, specifically, the manner in which a multidisciplinary design problem is posed and subsequently solved. Through the use of simulated cost functions, a more realistic outlook is gained as to the appropriateness and applicability of the various solution strategies with design systems of varying size and complexity. The next chapter continues with the global theme of this research effort – the reduction of time and cost within a multidisciplinary design – but on a more localized scale. Chapter 5 deals exclusively with techniques for converging the time and cost consuming system analysis, which is non-linear and hybrid-hierarchic for systems of real-world scope.

Chapter 5

Multidisciplinary Analysis Convergence

The previous chapter demonstrated how costly the MDO design cycle can be for simulated representations of real world engineering systems. By far, the costliest component of the MDO cycle is the system analysis, which for the MDF procedure, is highly iterative in nature. There are numerous, commonly used means for formal convergence of nonlinear multidisciplinary systems. Each will be discussed in detail, and shown to have numerous advantages and disadvantages. This chapter also presents the design and development of a new heuristic strategy for multidisciplinary analysis convergence, which hopes to build upon the strengths of the commonly used formal procedures. This chapter concludes with a lengthy comparison study of formal and heuristic convergence strategies, again using test simulations developed by the CASCADE system simulator.

Formal Analysis Convergence

a. Background

The "system analysis" depicted in Figure 2.7 ultimately translates into the solution of a large and complex system of nonlinear equations, in a "real world" engineering design. In such an instance, the best case scenario is that the closed-form

relations between subsystem outputs and subsystem inputs are known to the designer, complex as they might be. Often times, this exact relation is unknown to the managers of the design, and must be approximated through the use of some form of Response Surface Methodology (RSM) [8]. In either case, the numerical process of identifying a converged relation or "mapping" between output and input is often the most time and computationally cost-consuming portion of a multidisciplinary design process.

Through a quick survey of any Numerical Methods textbook, one clearly realizes that there are many numerical approaches for convergence of a nonlinear system of equations. Each has inherent advantages and drawbacks. Probably the two most widely used techniques for iterative convergence of nonlinear equations in MDO are Fixed-point Iteration [45] and Newton's Method [64]. An overview of these classical numerical approaches is presented as follows, starting with a presentation of the Fixed-point Iteration technique.

b. Fixed-point Iteration

Many nonlinear equations are formulated as fixed-point problems:

$$\mathbf{y} = K(\mathbf{y}) \qquad \textbf{[5.1]}$$

Here, K is the fixed-point mapping between the output and input, and is nonlinear in nature. A solution of [5.1] is denoted $\mathbf{y^*}$, and is called a "fixed-point" of the map K [46]. Fixed-point Iteration (FPI) is a convergence process that is denoted as follows:

$$\mathbf{y_{n+1}} = K(\mathbf{y_n}) \qquad \textbf{[5.2]}$$

In words, FPI is a zero-order method that attains convergence by successively substituting past estimates of a given variable towards the computation of a new estimate for the given variable. In a nonlinear system of equations, estimates of a given variable

are made based on past estimates of the variables of which the given variable is a function. This procedure gradually drives the variable in question towards convergence. Clearly, FPI is easily implemented and quite robust, but is in general, an inefficient means for numerical convergence.

c. Newton's Method

A more efficient, and extremely well-known alternative for nonlinear numerical convergence is called Newton's Method (NM). NM is a derivative-based scheme that results from the manipulation of a Taylor Series expansion about a design point y_o:

$$l(y) = f(y_o) + f'(y_o)(y - y_o) \qquad \textbf{[5.3]}$$

Analytically, it means that the linear function l(y) is close to the given function f(y) near y_o. At y_o, the two functions l(y) and f(y) agree [13]. The manipulated Taylor Series equation is then used to arrive at a new estimate for the variable in question as follows:

$$y_1 = y_o - [f(y_o) / f'(y_o)] \qquad \textbf{[5.4]}$$

When considering the simultaneous solution of a system of nonlinear equations, equation [5.4] is extended to the following:

$$\mathbf{y_{n+1}} = \mathbf{y_n} - F'(\mathbf{y_n})^{-1}F(\mathbf{y_n}) \qquad \textbf{[5.5]}$$

Here, $F'(\mathbf{y_n})$ is the well-known Jacobian matrix, and notice that its inverse is required to attain the new estimate of y.

NM is extremely efficient, but only when the initial design point is "sufficiently close" to the converged design point. If it is not, a different (and perhaps undesirable) root might be converged upon. Alternatively, outright divergence can often occur. As previously noted, NM also requires a matrix inversion, which is often computationally costly and difficult to achieve.

d. Formal Analysis Convergence – A Comparison

To illustrate these formal convergence concepts in the context of a multidisciplinary system, an example is presented. Figure 5.1 demonstrates non-serial coupling between subsystems A and B. Each subsystem requires independent input in the form of design variables X_A and X_B. In addition, each subsystem requires dependent input in the form of behavior variables Y_B or Y_A, each of which is an output quantity computed by the *other* subsystem. In other words, the computation of Y_A requires Y_B, and the computation of Y_B requires Y_A. For demonstration purposes, the following polynomial equations can be used to represent the structure of the coupling between subsystems A and B:

$$Y_A = 0.85X_A - 6.6e\text{-}09Y_B^3$$

$$Y_B = -0.00042X_B^2 + 0.000148Y_A^2$$

[5.6]

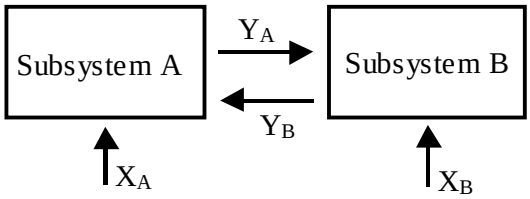


Figure 5.1: Two coupled subsystems

The equations are restated in an "equality constraint" form, whereby all non-zero variable values are moved to the right-hand side of the equation. Away from a converged design, the left-hand side values will equate to non-zero residuals. At convergence, these residuals are reduced to zero, and the following statements of equality will hold:

$$BVresid_A = 0 = 0.85X_A - 6.6e\text{-}09Y_B^3 - Y_A$$

$$BVresid_B = 0 = -0.00042X_B^2 + 0.000148Y_A^2 - Y_B$$

[5.7]

Note that the system is pre-defined by CASCADE to behave within a set of implicit bounds, [-9999, 9999]. In a real-world system, this restriction might be analogous to the presence of physical or geometric limits on the variables themselves.

The design variables, which are independent variables that remain constant during analysis convergence, have values of $X_A = 1123$ and $X_B = 2645$. A known solution for Y_A and Y_B is 1093.514 and -2761.355, respectively. Our pre-defined convergence criterion is that both equation residuals for Y_A and Y_B must be less than 0.01 at the end of iteration k, where the equation residuals are defined as follows:

$$residual(A) = abs(Y_A^k - Y_A^{k-1})$$

$$residual(B) = abs(Y_B^k - Y_B^{k-1})$$

[5.8]

Table 5.1: Formal analysis convergence

	Trial scenario #1				**Trial scenario #2**			
	FPI Y_A	**FPI Y_B**	**NM Y_A**	**NM Y_B**	**FPI Y_A**	**FPI Y_B**	**NM Y_A**	**NM Y_B**
0	600.	-2000.	600.	-2000.	-7834.	8321.	-7834.	8321.
1	1007.350	-2885.051	1070.823	-2801.432	-2847.961	6144.660	-11491.471	14625.837
2	1113.041	-2788.147	1093.443	-2761.455	-576.669	-1737.920	-10254.145	12396.915
3	1097.601	-2754.979	1093.517	-2761.355	989.194	-2889.114	-10004.617	11866.124
4	1092.556	-2760.031	1093.517	-2761.355	1113.711	-2793.512	-9991.681	11837.033
5	1093.317	-2761.666			1098.428	-2754.758	-9991.645	11836.948
6	1093.564	-2761.420			1092.523	-2759.762	-9991.645	11836.948
7	1093.526	-2761.340			1093.276	-2761.677		
8	1093.514	-2761.352			1093.565	-2761.433		
9	1093.516	-2761.356			1093.528	-2761.340		
10					1093.514	-2761.352		
11					1093.516	-2761.351		

For trial scenario 1, a design point "reasonably close" to the known converged solution is chosen: Y_A = 600, and Y_B = -2000. From this design, Table 5.1 shows that FPI converges in 9 iterations, and NM converges in 4 iterations. The iteration histories for Y_A and Y_B are seen in the Table.

For trial scenario 2, a design point that is "reasonably far" from the converged solution is chosen: Y_A = -7834, and Y_B = 8321. From this initial design, Table 5.1 shows that FPI converges in 11 iterations, and NM "diverges" to a different solution point after 6 iterations. This alternate solution is not necessarily incorrect. In fact, it is a secondary solution point that is *closer* to the initial point than is the "true" solution. However, the deficiency associated with this alternate solution point is that it does not lie within the pre-defined allowable [-9999, 9999] bounds. For this reason, this solution point must be deemed infeasible, and unusable.

The convergence strategy proposed in the present study attempts to build upon both the primary weakness of FPI (lack of efficiency) and the primary weakness of NM (lack of robustness with respect to starting design point). In so doing, it is hoped that a convergence strategy with "middle ground" characteristics (i.e. trade-off between robustness and efficiency) will result. This convergence strategy is referred to here as *Data Fusion Analysis Convergence* (DFAC) [41,44] from this point forward, and is described in detail in the next section.

Heuristic Analysis Convergence

a. Algorithm Overview

This section presents a discussion on the specifics of the DFAC technique. Before explaining details of the individual components of the algorithm, the overall flow of the method is described. Figure 5.2 demonstrates that the convergence strategy has four fundamental steps. First, each subsystem output (behavior variable) is modeled as a neuron of a Neural Network [35,59]. Upon initialization, the input/output relation of *each neuron* will have some degree of error, which must be minimized.

The second component of the convergence algorithm is the minimization (correction) of these errors, through the use of gradient-based optimization. Once the errors are minimized, each individual behavior variable equation will be satisfied. However, numerous behavior variables might require the same behavior variable as input, the values of which may not match at the end of the parallel optimization procedures taking place. Hence, a procedure for coordinating or "fusing" these discrepant numerical values is required.

This, in fact, is the third major step in the convergence algorithm – a procedure that is loosely based on the procedure of Data Fusion [33,53]. Finally, based on the results of the fusion process, a new predictive estimate for each behavior variable is calculated. This is the fourth and final step of the DFAC algorithm. The Figure 5.2 flowchart constitutes one cycle of the convergence algorithm, and is repeated numerous times until numerical convergence is attained.

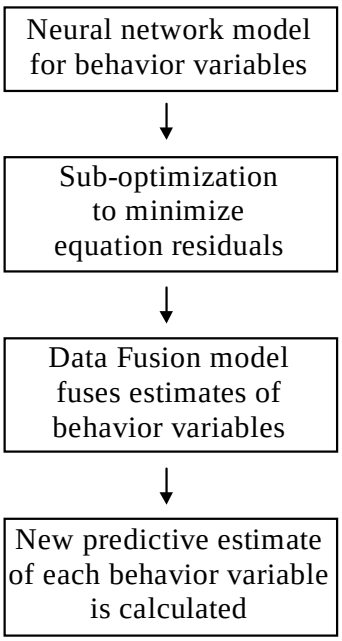


Figure 5.2: The DFAC process flowchart

As Figure 5.2 demonstrates, numerous optimization and MDO-based concepts have been combined to construct this new technique. Each of these background concepts is now presented, followed by a detailed discussion of how these concepts are used to form a useful new algorithm. The background discussion begins with Neural Networks, the structure of which is fundamental to the present research effort.

b. Neural Networks

In its most general form, a Neural Network (NN) is a "machine" that is designed to model the way in which the brain performs a particular task or function of interest [32]. NN's typically perform useful computations through a process of *learning*. To achieve good performance, NN's employ an interconnection of simple computing cells that are commonly referred to as "neurons". A neuron is an information processing unit that is fundamental to the operation of a NN.

The model of a neuron can be extended to an MDO context by realizing that each subsystem output (behavior variable) can be simulated as a neuron. The fixed inputs to each neuron are the subsystem design variables, which are known quantities, and which are held constant during the convergence procedure. The "non-constant" inputs to each neuron are the *input* behavior variables, whose values are unknown. A corresponding weight is defined for each input behavior variable, the values of which are initialized and to be determined. Hence, a full MDO model requires n concurrent neuron models, where n is the number of behavior variables in the multidisciplinary system.

On each iteration, the NN establishes a "computed" value for each of the behavior variables in the system that will likely be different than the corresponding input value for each. Hence, a sub-optimization problem is generated which will minimize the values of the computed errors for each behavior variable to equal zero. This is accomplished by altering the values of the weights corresponding to each input behavior variable, and is typically referred to as "error-correction learning" [70] in NN theory. The specifics of this sub-optimization problem will soon be discussed. First, the concept of Data Fusion is presented.

c. Data Fusion

The Neural Network-based approach simultaneously attains prospective solutions for each of the subsystem behavior variables. The problem is that *behavior variables are typically required as input by more than one subsystem*. Hence, multiple subsystem NN's will concurrently arrive at different values for the weights corresponding to the same behavior variable, despite the fact that at convergence, ALL weights corresponding to the same input quantity **must be equal**. As a result of this discrepancy, some means of

coordination must be devised to “blend” these non-equivalent weight values together to form a single intelligent estimate for each behavior variable. This process of blending is where data fusion comes into consideration.

Data Fusion techniques combine data from multiple “sensors” to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone [29]. Applications of multisensor data fusion are many. Historically, data fusion methods were developed primarily for military applications. Recent years have seen the development of numerous civilian applications for data fusion, including robotics and medical applications. Multisensory data fusion is naturally performed by humans to achieve a more accurate assessment of the surrounding environment. For example, a human cannot use vision alone to know what is around the corner of a wall. However, the *combined* use of vision, hearing, and sense of smell can provide a greater understanding of what is around that same corner. Clearly, this ideology can be extended and implemented in a numerical/computational context – namely, for the multidisciplinary analysis convergence problem at hand.

The impetus for the incorporation of Data Fusion concepts into MDO stems from a preliminary investigation conducted during the summer of 1997, which focused on increasing the efficiency of the entire MDF solution cycle by eliminating the requirement and computational overhead of assigning design variables to individual subsystems. The present approach also exploits sensitivity information to guide the convergence process but diverts the attention away from the optimization cycle, and design variable allocation altogether. The present effort utilizes the data fusion concept for intelligently blending discrepant behavior variable information computed concurrently by different subsystems

during iterative analysis convergence, in order to increase the efficiency of this costly process.

With all of the background components described, the specifics of the DFAC algorithm can now be discussed.

d. DFAC Algorithm

The algorithm begins by modeling each subsystem output (behavior variable) as a separate, concurrently executed neuron model (see Figure 5.3, which is divided into four "segments", (a), (b), (c), and (d)), each of which is now described.

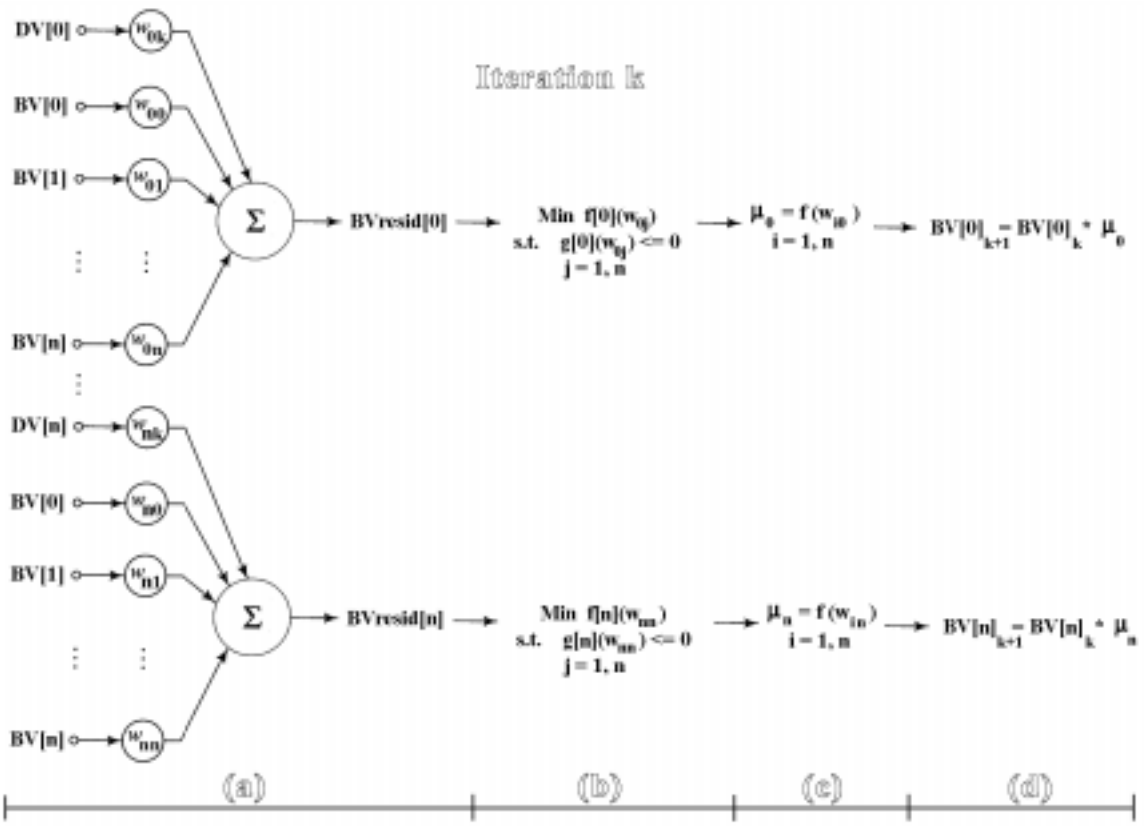


Figure 5.3: The DFAC algorithm

Segment (a): Each neuron has fixed input in the form of design variables, and "non-fixed" behavior variable inputs with corresponding variable weights. There is one neuron for each "output" behavior variable. Note that each behavior variable equation is

re-stated in "equality constraint" form as was demonstrated in equation [5.7]. Hence, unlike the FPI and NM convergence implementations, the representative equation corresponding to each behavior variable need not be separable, which could be a tremendous advantage. Because of this, each output behavior variable quantity also serves as an input to the neuron, and has an associated weight, in the NN equation model. This implies that the desired output of each neuron - the behavior variable equation *residual* - is zero.

Segment (b): To reduce the residuals to zero, a sub-optimization problem is created. Each neuron has an associated objective function and an associated equality constraint function. The "duty" of the objective function is to keep the values of the weights as close to their initial values as possible. The "duty" of the constraint function is to ensure that at the end of the optimization cycle, the behavior variable equation residual is equal to zero. The forms of the optimization functions used for achieving error minimization are given as follows:

$$\text{Minimize:} \quad F = \Sigma\, C_O(C_I(x_i - \mu_i))^2 \quad i = 1, n$$

$$\text{subject to:} \quad g = \Sigma\, D_O(D_I(BVresid_i))^2 \quad i = 1, n$$

$$x_{lb} \le x_i \le x_{ub} \quad i = 1, n$$

[5.9]

Here, C_I, C_O, D_I, and D_O are user-defined constants, x_i is the fusion weight corresponding to behavior variable i, $BVresid_i$ is the equation residual corresponding to behavior variable i, and μ_i is the fusion estimate value corresponding to each fusion weight, which was computed on the previous iteration. Clearly, n is the total number of behavior variables that require convergence. The fusion weights are assigned to behave within

some set of lower and upper bounds, x_{lb} and x_{ub}, respectively, which are user-defined. The Automated Design Synthesis (ADS) [85] optimizer has been implemented to handle the sub-optimizations. For all example problems presented in this study, an exterior penalty function strategy is used, along with a DFP variable metric search, and a combined Golden Section / Polynomial interpolation 1-D search.

Segment (c): Once all NN residuals have been reduced to zero, a data fusion procedure is required. This is because each NN will arrive at conflicting values for the weights that correspond to the *same* behavior variable. A converged system analysis occurs when all weight "instances" corresponding to a given behavior variable are equal, for all system behavior variables.

Four data fusion models have been developed in the present research effort. The data fusion technique that has had the greatest success thus far is referred to as "derivative-based" fusion. In this scheme, derivatives of each equation residual are taken with respect to each behavior variable weight found in that respective equation. These derivative quantities are denoted "BVsens" in equation [5.10] below. Recognize that this is analogous to taking the derivative of an equality constraint equation with respect to its design variables. Here, the weights associated with a behavior variable equation whose sensitivity was found to be the largest will be weighted proportionally larger than those of a behavior variable equation whose sensitivity was found to be smaller. The relationship developed here is written:

$$\mu_i = \frac{\sum_{j=1}^{n} (w_{ij} * (BVsens)_j)}{\sum_{k=1}^{n} (BVsens)_k} \qquad i = 1, n$$

[5.10]

Here, "i" represents the fusion variable for which this computation is taking place, "j" and "k" are generic indices, and "n" is the total number of behavior variables (output neurons) in the system. Clearly, "w_{ij}" is the weight corresponding to a coupling term in equation i that is coupled to another subsystem, by virtue of behavior variable j.

Before proceeding, the three alternate data fusion models that have been developed in this research effort are presented. The simplest of these is called "mean-based" fusion. As the name implies, the fusion estimate for a given behavior variable is based solely on the numerical mean of **all** weights corresponding to that variable. This relationship is represented as follows:

$$\mu_i = \frac{\sum_{j=1}^{n} (w_{ij})}{p} \qquad i = 1, n$$

[5.11]

Here, all parameters are as they were initially defined in equation [5.10]. Quantity "p" is an integer, and represents the total number of behavior variable equations which are a function of behavior variable i. Note that the weight w_{ij} corresponding to a given behavior variable i of which behavior variable equation j is NOT a function will have a value of 0.0.

A slightly more sophisticated fusion algorithm is called "residual-based" fusion. This algorithm gives precedence to weights that correspond to couplings that correspond

to behavior variables that are "farthest away" from being converged, this having the largest equation residual, "BVresid", on a given convergence iteration. This relationship is represented as follows:

$$\mu_i = \frac{\sum_{j=1}^{n} (w_{ij} * (BVresid)_j)}{\sum_{k=1}^{n} (BVresid)_k} \qquad i = 1, n$$

[5.12]

Finally, the most sophisticated fusion model is presented. However, note that this model did not quite exhibit the success of the "derivative-based" model during preliminary testing. This final model attempts to blend the concepts of equation [5.10] and [5.12] and is called "derivative/residual-based" fusion. This relationship is represented as follows:

$$\mu_i = \frac{\sum_{j=1}^{n} (w_{ij} * (BVresid)_j * (BVsens)_j)}{\sum_{k=1}^{n} ((BVresid)_k * (BVsens)_k)} \qquad i = 1, n$$

[5.13]

The "derivative-based" model seen in equation [5.10] is used as the fusion model for all results attained in this chapter, due to its success during numerous preliminary test cases. It is suspected that the other three fusion models are still useful in certain situations, and deserve future consideration. Their general forms have hence been provided for sake of completeness.

Segment (d): Once a "fused" weight is calculated for each of the system behavior variables, a new estimate of each behavior variable is calculated. This is accomplished

by multiplying the input value of a given behavior variable (on a given NN cycle) by the corresponding calculated fused weight, for all system behavior variables:

$$BVnew_i = BVold_i * \mu_I \ , \qquad i = 1, n \qquad \textbf{[5.14]}$$

These new estimates on analysis cycle "k" are compared to their corresponding values on analysis cycle "k-1". If, for all behavior variables, the difference between "BV_k" and "BV_{k-1}" is less than some convergence threshold, then the DFAC algorithm has converged. If not, the entire process repeats itself until convergence is achieved, or until some predetermined number of cycles are executed.

The DFAC algorithm is not wholly deterministic, as some additional parameters contribute to the convergence procedure which contain some degree of uncertainty. These include a [0,1] "randomness factor" which pre-multiplies the relative finite difference coefficient in the optimizer. In addition, the user can assign a "decimal improvement" parameter, which decides the degree of improvement that must be attained on each iteration of the convergence procedure to accept the new fusion-generated move. A default value of 1.0 implies that any reduction in the sum of the behavior variable residuals over the corresponding sum (on the previous iteration) will be accepted. A value of 1.25 implies that the residual sum would be allowed to *increase* by as much as 25%, and the data fusion-based move would still be accepted. Hence, DFAC acts like certain heuristic optimization techniques such as Simulated Annealing [48]. Namely, there is some probability of accepting a design point with an "inferior" solution alternative, in hopes of escaping any local minima that reside within the solution space.

The DFAC algorithm has been compared with the formal convergence strategies through the implementations of two different forms of coupled test problems. These test problem classifications are now described.

Simulation Details and Results

a. Test Problem Descriptions

In this research effort, three different fully-coupled test systems have been generated by CASCADE, each having three subsystems, and one behavior variable (output) per subsystem, and one design variable per behavior variable. Figure 5.4 illustrates the coupling structure of all three test systems, each of which will have differing semantic characteristics. The specific equations for these three test systems are found in Appendix I.

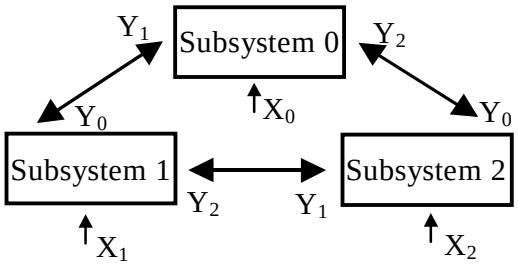


Figure 5.4: CASCADE test system structure

Though the use of CASCADE-generated test systems is seminal for this research effort, a secondary means for testing the DFAC algorithm is desired. These secondary problems should be less generic in nature, and should demonstrate characteristics of a "real-world" MDO problem. Moreover, these problems would ideally be a bit larger in size, and not necessarily fully coupled in nature. The obvious choice for the source of such problems is the MDO Test Suite at NASA Langley [63]. For this effort, two problems from Class I were chosen -- the "Heart Dipole (HD)" and the "Combustion of Propane (COP)" problems. Both problems are, in their simplest form, a series of n

nonlinear equations and n unknowns, where n is equal to 8 for the HD problem, and 11 for the COP problem. Figures 5.5 and 5.6 illustrate the coupling structure of the HD and COP systems, respectively.

These test problems have been modified slightly from their original form so that they could be used for the research purposes of the present effort. The systems of equations were modified to be separable in each of the n unknowns, which allowed for the comparison of the DFAC algorithm to formal convergence techniques that require separability. Further modifications have been made such that the equations demonstrated some degree of stability from numerous starting points. The modified versions of these equations are seen in Appendix II.

b. Hybrid Strategy Convergence Approach

The approach used for attaining all results in this study is a "hybrid" strategy. By this is meant the following: the DFAC algorithm is first implemented to reduce the initial equation residuals down to a near-zero value. Thereafter, the FPI convergence technique is implemented to finish off the convergence process. Clearly a large research issue that stems from the current effort is the ability to identify the threshold point at which the DFAC algorithm is "turned off" and FPI is "turned on" to finish the job. For the present research effort, it was discovered through preliminary testing that in general, an appropriate number of preliminary iterations of the DFAC algorithm is to be approximately k/2. Here, k is the number of iterations required to converge the system using straight FPI (for every iteration) from the given starting point.

Note that NM or another formal convergence algorithm could be used to complete the convergence process, instead of FPI. FPI has been chosen for the acquisition of the results in this study, mainly for its high level of robustness.

c. Results

Table 5.2 provides the following information for each of the five test systems. First, the initial sum of equation residuals (row a), and the final sum of equation residuals after application of the DFAC algorithm (row b). Next is the number of applied iterations of the DFAC algorithm (row c), and the subsequent number of FPI iterations required to fully converge the system to one decimal place (row d). Finally, the total number of hybrid iterations (row e) required for convergence is shown. For reference, the number of iterations required to converge the same system using the FPI (row f) and the NM (row g) techniques are also supplied. Note that *italicized* rows b, c, d, and e apply to the DFAC algorithm.

Table 5.2: Result summary

	CASCADE 1	**CASCADE 2**	**CASCADE 3**	**HD**	**COP**
a. Initial residual sum	10.27	9.51	29013.76	6.194	18.996
b. Final residual sum	0.007	0.012	0.111	0.031	0.084
c. Iterations of DFAC algorithm	15	12	35	25	20
d. Subsequent FPI iterations	15	6	1	12	34
e. Total "hybrid" iterations to	30	18	36	37	54
f. FPI iterations to converge	37	24	77	60	72
g. NM iterations to converge	diverges	7	5	7	diverges

Figure 5.5 is a chart that summarizes the results for all five test systems side by side. The convergence techniques are abbreviated FPI, NM, and DFAC. Plotted on the y-axis is the number of iterations required to converge the system to the appropriate number of decimal places, as outlined in the previous sub-sections. Note again that NM

diverges (denoted by "∞") for the first CASCADE system and for the MTS COP system. A discussion of these results follows.

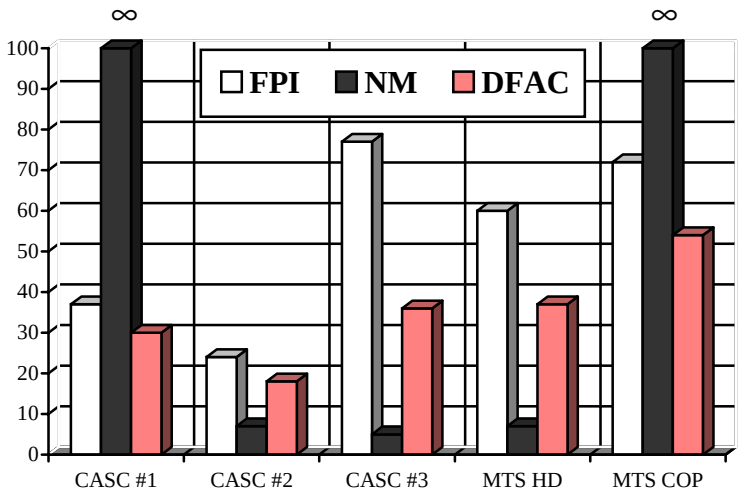


Figure 5.5: Comparison of convergence strategies

Discussion

The DFAC algorithm has proven successful for all three CASCADE-based test systems. Here, "success" is measured by the fact that straight FPI iteration was bettered by the DFAC algorithm by an appreciable amount for every test case. For test system #1, 15 iterations of the DFAC algorithm were implemented, followed by 15 more iterations of straight FPI to converge the system. Hence, this hybrid strategy converged the system in a total of 30 iterations - 7 iterations (18.9%) fewer than straight FPI. Recall that NM diverged from the given starting point, for the first test system. For test system #2, 12 iterations of the DFAC algorithm were implemented to reduce the initial residuals, followed by 6 iterations of straight FPI to achieve full convergence. Hence, this hybrid strategy converged the system in a total of 18 iterations - 6 iterations (25%) fewer than straight FPI. NM converged in a mere 7 iterations for this test system. Finally, for test

system #3, 35 iterations of the DFAC algorithm were implemented to reduce the initial residuals, followed by only 1 iteration of straight FPI to achieve full convergence. Here, it is clear that the DFAC algorithm nearly converged the system on its own accord. Ultimately, the hybrid strategy converged the system in a total of 36 iterations - 41 iterations (53.2%) fewer than straight FPI. NM converged in a mere 5 iterations for this test system.

A large degree of success was also seen for the two "non-CASCADE" systems, though after many experimental hours of testing and re-testing the algorithm. For the modified HD system, 25 iterations of the DFAC algorithm were implemented to reduce the initial residuals, followed by 12 iterations of straight FPI to achieve full convergence. Hence, this hybrid strategy converged the system in a total of 37 iterations - 23 iterations (38.3%) fewer than straight FPI. NM converged in a mere 7 iterations for this test system. For the COP system, 20 iterations of the DFAC algorithm were implemented to reduce the initial residuals, followed by 34 iterations of straight FPI to achieve full convergence. Hence, this hybrid strategy converged the system in a total of 54 iterations - 18 iterations (25.0%) fewer than straight FPI. NM diverged for this test system.

After preliminary testing, a few general observations can be made. The DFAC algorithm is seemingly very good at reducing large equation residuals down to values close to zero, where straight FPI can then be implemented to drive the near-zero residuals down to zero. The general trend seen in the experimental testing is that for "accurate" starting points that turn out to be "close enough" to the true solution, a formal derivative based convergence scheme such as NM is clearly the convergence strategy of choice. NM is reliable in cases where the analysis solution space is known to be convex. In

many realistic MDO applications, this will rarely be the case. Alternatively, when the starting point is known to be a "substantial distance" from the converged solution, or if a relation between the initial and final solutions cannot be estimated (proving NM to be an unsafe choice for iterative convergence), the DFAC algorithm has shown itself to be reliable for reducing large residuals. DFAC can then be supplemented with numerous iterations of FPI to achieve full convergence. This hybrid strategy has shown to be more efficient than straight FPI in all five test cases.

The DFAC algorithm involves numerous concepts and procedures, and is clearly more complicated and computationally intensive than is FPI. The ideal application of this technique is envisioned to be a large-scale MDO problem, where iteration time is known to be substantial (i.e. on the order of "hours/days" as opposed to "seconds/minutes"). Hence the implementation of the DFAC algorithm will see an increase in time and computational cost per iteration, in the hopes of decreasing the total number of iterations to attain a converged design. NM is clearly more computationally intensive than FPI, as it requires derivative computations and a matrix inversion, which is typically a high cost endeavor. It is suspected that the computational cost of the DFAC algorithm is also larger than that of the NM implementation. While the DFAC results have not yet shown to be as efficient as those attained by NM, the technique has shown to exhibit the robustness of FPI, which is a large shortcoming of NM.

The last two chapters have discussed, in great detail, specific applications for the use of the CASCADE simulations to identify various means for time and cost reduction of the elaborate MDO design cycle. Chapter 6 presents the design and development of

FACETS, the MDO framework that encompasses all of the independently developed areas of MDO research that have been presented in the present work.

Chapter 6

MDO Framework design and development: FACETS

The current chapter discusses the design and continual development of a new computational framework in the field of MDO, entitled FACETS. The primary purpose of FACETS is to bring together the numerous MDO tools and techniques that are available to a design manager into a single, all-encompassing infrastructure. Such a tool can provide an MDO design manager with a powerful means for identifying possibilities for time and cost reduction within an existing multidisciplinary design. There are numerous distinctions between FACETS and other MDO frameworks, and each will be discussed within the body of this chapter. The primary difference is that FACETS is a preliminary design tool, *rooted in simulation*; hence, CASCADE is a vital feature in this framework. Prior to presenting a lengthy discussion on the FACETS framework, a literature survey and background discussion of existing, large-scale MDO computer tools is presented.

Literature Survey and Background

Clearly, as the preceding chapters of this dissertation have demonstrated, there are many research avenues within the field of MDO which ultimately strive to address to very same research issue – the reduction of time and cost within the multidisciplinary

design cycle. To address this seminal issue, numerous computational problem-solving environments, commonly referred to as “frameworks”, have been developed in recent years. A framework has been loosely defined in literature as a “hardware and software architecture that enables integration, execution, and communication among diverse disciplinary processes” [69].

Some recently developed frameworks in MDO which incorporate some combination of these characteristics include FIDO (Framework for Interdisciplinary Design Optimization) [87], which serves as a general distributed computing system for executing multidisciplinary computations on a heterogeneous network or workstations. FIDO automates the coordination of analyses by numerous design disciplines into an integrated optimization scheme, and allows for visualization and “steering” by the designer.

A number of other frameworks and/or problem solving environments are presently under development. These range from alternatives to FIDO which focus more on exploiting distributed, heterogeneous computing, such as Access Manager [66], developed by Boeing. Others include commercial optimization toolkit environments such as iSIGHT [82], which allows the user to flexibly integrate analyses with optimization methods of all forms; numerical, heuristic, design of experiments (DOE) [58], and others. A similar optimization toolkit environment is LMS-Optimus [24], which provides the user with DOE and RSM methods in addition to conventional nonlinear programming techniques.

Finally, there are other existing design tools that focus more on data and information flow within the design process, such as IMAGE [28], which provides object-

oriented data management utilities for use during design processes. Another such example is the DARWIN [49] project, which seeks to reduce design cycle time by improving access to experimental data.

In reference [71], Salas and Townsend outline four of the essential requirements that a viable MDO framework must contain. These include, but are not limited to the following aspects:

1. First, *architectural design* - the framework should have an attractive front-end, such as an intuitive Graphical User Interface.
2. Second, *problem formulation construction* - the framework should support legacy and proprietary codes.
3. Third, *problem execution* - the framework should automate the execution of processes, and allow for parallel execution and user interaction.
4. Finally, *information access* - the framework should provide visualization and monitoring capabilities.

The primary shortcomings of many existing MDO frameworks are that they tend to be hard-coded, discipline or problem specific, and have limited capabilities when it comes to the incorporation of new technologies. There appears to be a need for a framework which can exploit many of the newly developed tools, strategies and techniques in MDO which strive to “simplify”, and ultimately reduce the time and cost of the design cycle associated with large, coupled engineering design problems.

Many of these tools and techniques have been outlined in the preceding chapters of the present work. These and other computational techniques can be viewed upon as “islands of research” in that they are independently developed computer codes and

concepts, which are (at present) physically separated, yet functionally related [42]. A design manager may benefit from an environment which allows the combination and/or integration of such research islands, such that related research concepts can be merged, and numerous "what if?" scenarios can be explored quickly and easily. An inter-related problem involving many of these research islands, and MDO research in general, is the lack of availability of design data and benchmark (test) problems. Researchers must have a safe and robust means for testing a newly developed MDO strategy prior to its implementation on an actual "real-world" design. Hence, the development of a framework that is focused on the incorporation of new MDO tools and techniques should have a robust means for coupled system simulation, both at the system analysis and optimization levels, as its foundation.

Preliminary Conception

The MDO Framework that is presently under development is called FACETS [42], which is an acronym that stands for "Framework for the Analysis of Coupled Engineering Techniques in Simulation." The original concept for this research effort was to utilize the Java programming language [19] to program the exterior structure of the framework. Java is an interpreted programming language whose compiler uses byte-code rather than native machine code. In doing so, the framework would have taken strides towards being heterogeneous; in other words, capable of execution on any architecture that has a Java interpreter. Java programs that are written as applets can be executed through the use of a web browser, which are typically available on all computational platforms (PC/Macintosh/Workstation) and operating systems. Unfortunately, the shortcomings of using Java as a large-scale MDO tool are numerous; namely, there are

security limitations on implementing Java applets for both reading and writing files, and instantiating system-level commands. These are operations that are fundamental to any large-scale software tool that relies on inter-communication between related computer codes and functions. These weaknesses, coupled with Java's slow execution speed in the context of numerical computations deemed the use of the language infeasible for the task at hand.

Motif

As a result of the numerous shortcomings of Java, the exterior front-end of the FACETS framework has a Graphical User Interface (GUI) that has been coded using the Motif toolkit [34]. Motif, which utilizes the ANSI C programming language, was designed by the Open Software Foundation (OSF). The OSF is a consortium of companies such as Hewlett-Packard, IBM, Digital, and others, whose charter calls for the "development of technologies that will enhance interoperability between computers from different manufacturers" [34]. This line of thinking is clearly parallel with the necessary aspirations of any useful MDO computer tool.

Motif is based on the X-Windows System, which is a network-based windowing system that has been implemented for UNIX, DOS, Macintosh, and other operating systems, and serves as a flexible foundation for GUI-based programming. Each of the modules within FACETS implements the easy to use control "widgets" that are typical of all Graphical User Interfaces. These include push buttons, scrollbars, toggle (radio) buttons, textfields, and others. A generic flowchart of the structural operability of FACETS is seen in Figure 6.1. Note that the modules marked with an asterisk (*) are "future modules" that are not yet incorporated into FACETS.

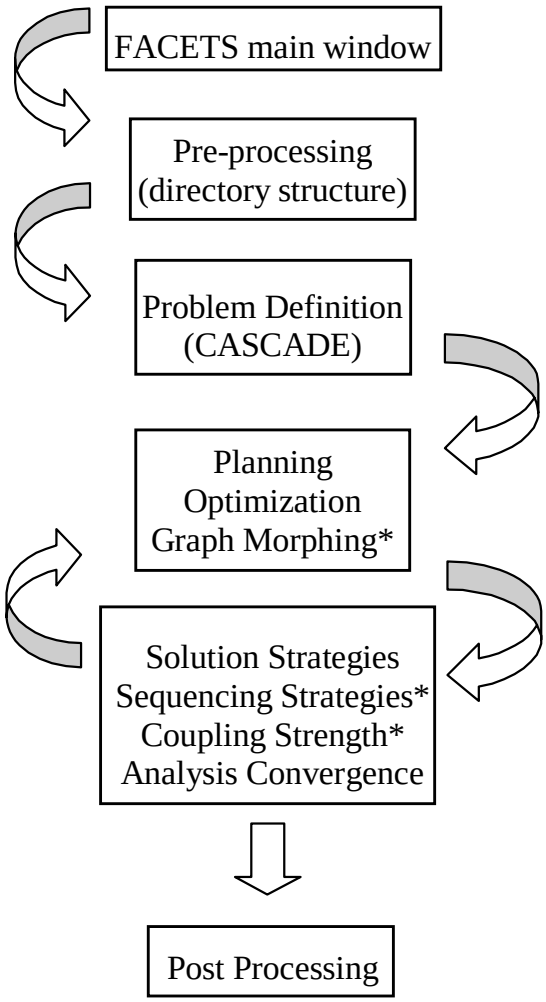


Figure 6.1: FACETS general structure

When the user instantiates FACETS, the main GUI window appears as shown in Figure 6.2. Hereafter, the user clicks on the FACETS name, which is in fact a colored button. This event then brings up a window that allows the user to establish a directory structure; this window is shown in Figure 6.3. Here, the user utilizes the textfields to prescribe the location of file paths for each module. This is necessary because FACETS makes use of numerous modules, each of which requires a number of input and output files for successful operation. It is for this reason that this "directory structure" window

is the first module presented to the user. This information must be established prior to entering any of the feature modules of FACETS.

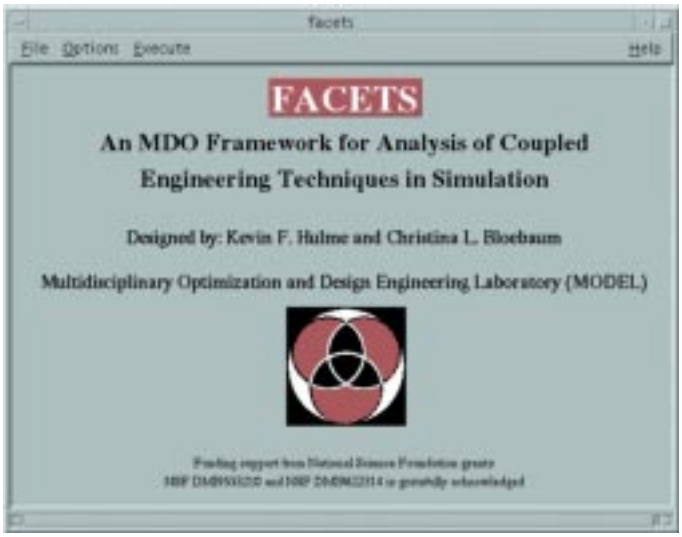


Figure 6.2: FACETS main window

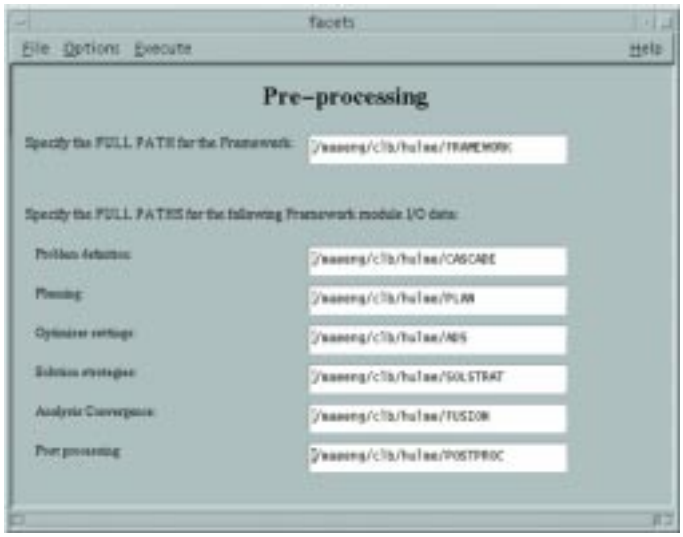


Figure 6.3: FACETS directory structure window

As hinted upon previously, FACETS operates upon the concept of "modularity". In other words, each application or feature of FACETS functions as its own individual entity. This flexibility allows for the addition of new technologies to, or the removal of dated applications from the framework at any time. The modules within FACETS can then "communicate" through numerous means. Typically, simple numerical data is

stored in data files that can be written as output by one module, and read in as input by another module. For example, the final values of the constraint functions are written to a data file by a given module, and are then read in by the post-processing module for visual presentation.

Some modules create output in the form of a compilable language file. For example, the analysis equations generated by the CASCADE simulator are written in the form of compilable ANSI C based functions, and are all encapsulated within a single header file. At present, all output files are written in ANSI C, but this process could easily be extended to other languages, such as Fortran, Java or even HTML [60]. These language files can then be compiled with other codes and fully used by other modules.

Any "command line" activity is accomplished by FACETS via the UNIX-based "system" command:

system("command"); **[6.1]**

where "command" is a character string in the form of an operating system command. For example, should the user wish to compile two files "out1.c" and "out2.c" that are located in the /tmp directory, and subsequently write the executable to a file named "out" located in the same directory, the system command might appear as follows:

system ("cc –o /tmp/out /tmp/out1.c /tmp/out2.c"); **[6.2]**

The system command is used primarily for compiling and executing codes, and is triggered through a widget event, namely the depression of a push button.

Some additional general features of the FACETS framework are worthy of mention. The user has the power to interrupt the execution of a module prior to its completion. This might be desirable if the user notices that the solution process is

proceeding into a disadvantageous region of the solution space. Eventually, FACETS will allow the user to make some form of an adjustment during the interrupt period, and then restart execution after this alteration has been made. An additional feature involves data storage. The user may wish to save or "store" result data after an execution of FACETS, which can later be retrieved and "opened" for subsequent usage. This process can take place before or after the execution of any of the feature modules in FACETS.

The next section addresses some of the specific modules that have been or will soon be incorporated into the FACETS framework structure.

Modules of FACETS

This section discusses all of the modules that appear in the "first release" of the FACETS framework. The modules that have already been developed by the authors are emphasized. It is envisioned that the incorporation of future modules will include MDO strategies developed by exterior researchers.

a. CASCADE

After instantiating the framework and establishing a directory structure, the user must define the multidisciplinary problem data. The long-range goal for this framework is to accommodate "real world" design data in addition to simulated design data. At present, the framework serves solely as a preliminary design tool. Hence, problem data is artificially generated using the CASCADE simulator.

The FACETS framework contains a "Problem Definition" module that houses the CASCADE simulator. The corresponding FACETS window is shown in Figure 6.4. This module contains three sub-modules. The first is the "system analysis" sub-module, which allows the user to define the structure of the analysis equations to be generated. As

shown in the figure, the user manipulates simple widgets to define the number of subsystems, design variables, and behavior variables in the system to be generated. Additionally, the user assigns parameters relating to convergence, system volatility, and others.

The second is the “optimization” sub-module, which is triggered by clicking the corresponding radio button near the center of the window. Here, the user assigns options pertaining to the optimization portion of the simulation, such as the number of inequality constraints per subsystem, and the nature of the objective function (system or subsystem-level).

The third is the “evaluation cost” sub-module, and is triggered by pressing the corresponding radio button near the right-center of the window. It is here that associated generic evaluation costs are defined for both the analysis behavior variables and for the objective function and inequality constraint functions. Such information is useful when comparing multidisciplinary solution strategies, as was discussed in Chapter 4.

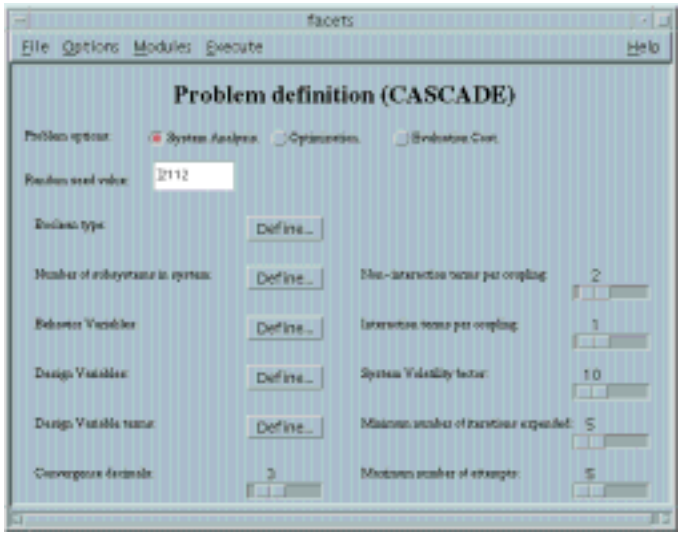


Figure 6.4: FACETS “Problem Definition” module

The FACETS framework is primarily interested in testing MDO methods and strategies on a simulation-based level. It is because of this that CASCADE, which ultimately provides the multidisciplinary problem data, is truly the flagship module of the framework.

b. Planning

The development of the Planning module was inspired by a similar (and much more elaborate) module found in Rogers' DeMAID [67] program. This module allows the user to visualize initial information about the problem being solved. Currently, this "pre-processing" module serves two primary functions. The first is to allow the user to visualize the initial value of the system objective function (or the summed value of the subsystem objective functions). This information provides a baseline measure to quantify future design improvements against. Recall that CASCADE generates the optimization simulation around the system analysis in such a way that the converged starting point is a feasible one. At present, the user is given the option to seek out an alternate initial feasible starting design point by way of a random search. Such a procedure could easily be expanded to allow for a more elaborate search procedure, as well. This sub-module of the Planning module is seen in Figure 6.5a.

The second primary feature of the Planning module involves the coupling structure of both the system analysis and optimization equations generated. Namely, this sub-module informs the user as to whether or not there are a). behavior variables which are not coupled (i.e. are not required as input by other behavior variables), or b). behavior variables which do not require any other behavior variables as input. Analogous information is provided for the optimization functions as well. Namely, are there c).

behavior variables which are not coupled (i.e. are not required as input by any optimization functions), or d.) optimization functions which do not require any behavior variables as input. Such information could be useful when comparing strategies for multidisciplinary analysis and optimization. This sub-module of the Planning module is seen in Figure 6.5b.

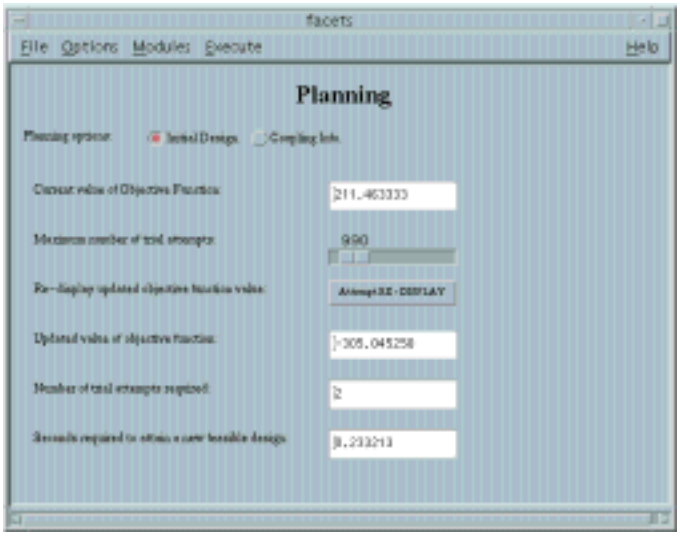
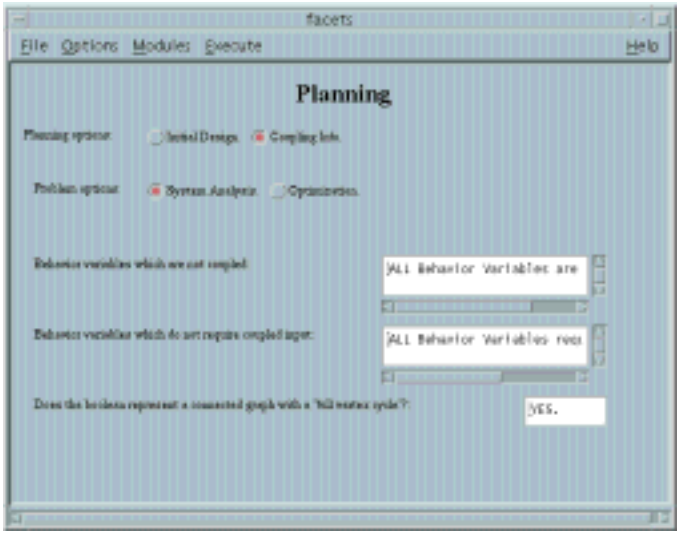


Figure 6.5: FACETS "Planning" module
a. Initial design point



b. Coupling information

c. Optimization

The FACETS framework presently makes use of the Automated Design Synthesis (ADS) program [85] as its sole optimization tool. ADS allows the user to change a large number of options corresponding to the gradient-based optimization search. Such parameters relate to the optimization strategy, optimization method, the one-dimensional search method, move limits, constraint thickness, finite difference parameters, variable scaling, and many others. A module has been devoted to providing the user with an easy means for assigning and altering such options quickly and easily. This allows for the efficient execution of "what if?" scenarios within the context of any MDO techniques that require numerical optimization. The "optimizer settings" module main window is shown in Figure 6.6.

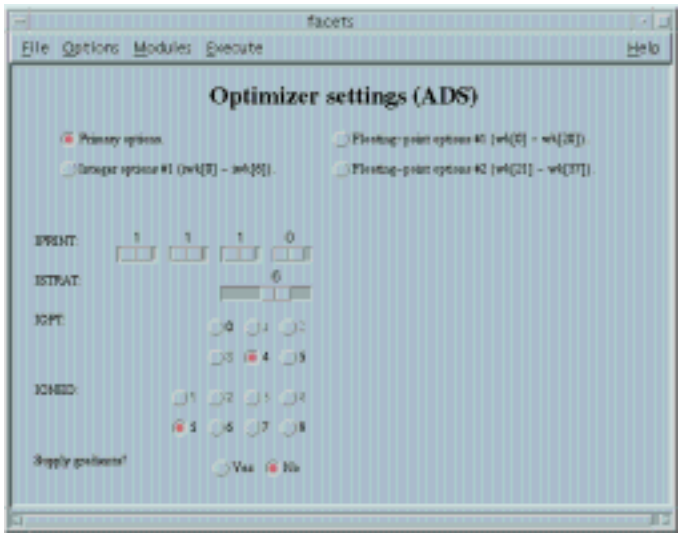


Figure 6.6: FACETS "Optimization" module

d. Solution Strategies

As discussed in Chapter 4, there are numerous means for posing and subsequently solving a multidisciplinary design problem. The most popular and well-known strategy has been called Multiple-Discipline Feasible (MDF). Two of the more popular alternate

strategies are called All-at-Once (AAO) and Individual-Discipline Feasible (IDF). An entire FACETS module has been created which allows the user to compare these solution strategies within the context of the same multidisciplinary design problem. Depending on the strategy chosen, there are numerous implementation issues to be considered. For example, if an explicit system analysis is utilized in the solution strategy chosen (as in both MDF and IDF), there are numerous analysis convergence-related options to be assigned. Also, if there are additional optimization variables introduced in the solution strategy chosen (as in both IDF and AAO), corresponding side constraints must be defined. As a baseline solution strategy, Random Search [91] is also included as an available multidisciplinary solution strategy. The main window for Solution Strategy assignment is shown in Figure 6.7.

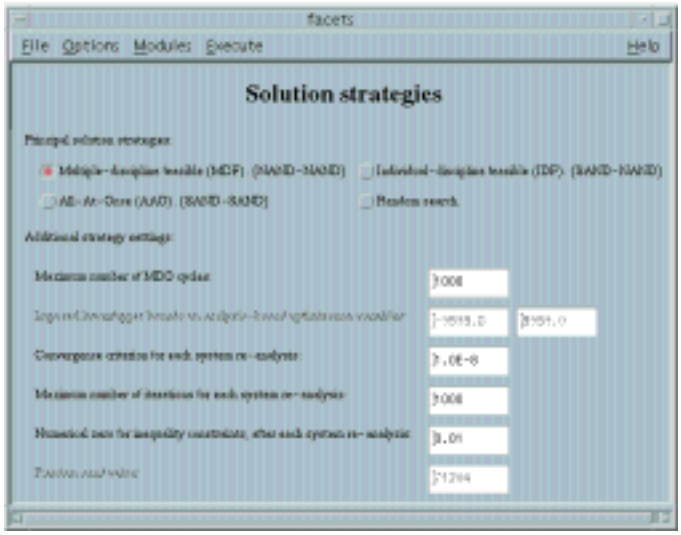


Figure 6.7: FACETS "Solution Strategies" module

e. Analysis Convergence

As discussed in Chapter 5, the system analysis tends to be the most costly component of the multidisciplinary design cycle. Hence, an entire module has been devoted to allow for the comparison of strategies for the convergence of a

multidisciplinary analysis. These range from well-known formal convergence strategies such as Fixed-point Iteration that is based upon successive substitution, and Newton's Method, which is derived from a Taylor Series expansion. Both of these techniques are available as analysis convergence options within FACETS. Each of these techniques has different strengths and weaknesses. As a result of this, the DFAC convergence algorithm presented in Chapter 5 is also available within FACETS as a possible convergence alternative. Four "fusion models" have been developed thus far, all of which are available as options within the FACETS analysis convergence module. The main window for Analysis Convergence assignment is shown in Figure 6.8.

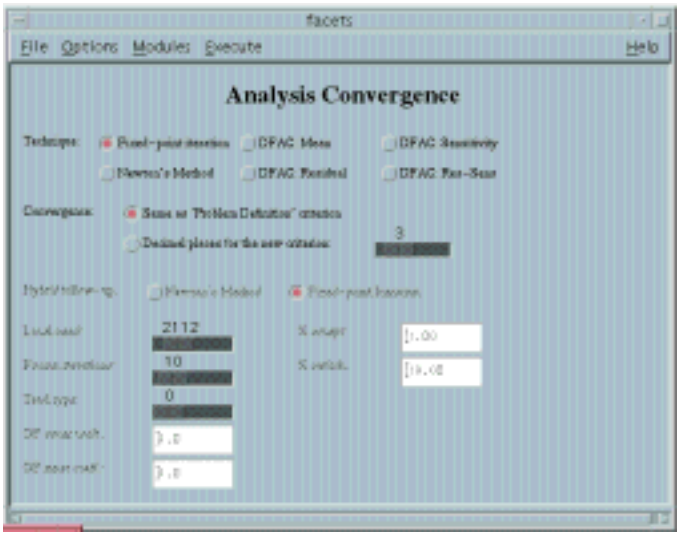


Figure 6.8: FACETS "Analysis Convergence" module

f. Post Processing

Finally, a means for quantifying results is imperative in any multidisciplinary design framework. FACETS post-processor provides the user with a variety of means for result interpretation and visualization. These include an "optimization feedback" sub-module, which allows the user to view the final value of the objective function, as

well as a color-coded means for visualizing constraint status (active/violated/satisfied). This sub-module is shown in Figure 6.9a.

The “time and cost” sub-module allows the user to view the overall CPU time and the generic cost associated with the multidisciplinary solution cycle. This sub-module is shown in Figure 6.9b. Finally, a “plotting” sub-module uses the XMGR [83] UNIX-based routine to provide the user with 2D plots of iteration convergence at the analysis level, objective function convergence at the MDO-cycle level, and design and behavior variable histories at both the analysis and MDO-cycle levels. This sub-module is shown in Figure 6.9c.

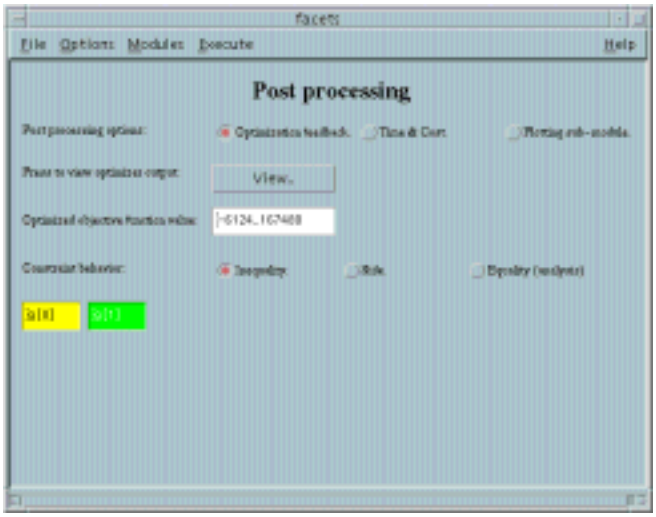
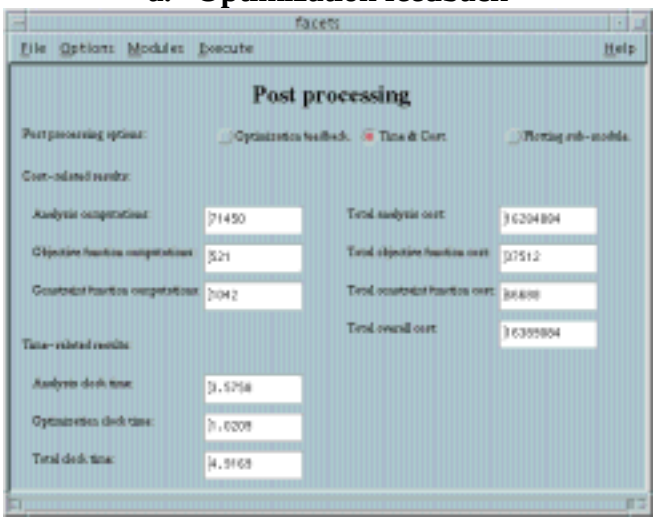
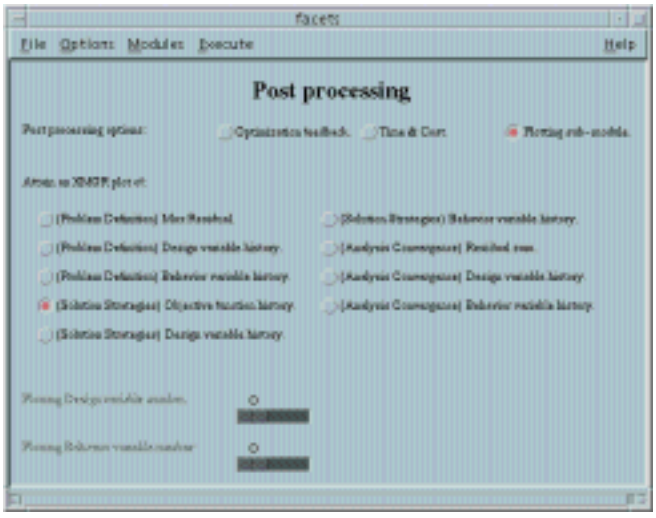


Figure 6.9: FACETS "Post Processing" module

a. Optimization feedback



b. Time and cost



c. Plotting

The framework has been used for numerous MDO applications. The next section presents a verbal demonstration run through of FACETS, where the "Solution strategies" module is the primary feature utilized. In so doing, the utility of most of the current features of FACETS is demonstrated.

Demonstration Usage of FACETS

In this demonstration, the user wishes to gain insight as to the overall MDO solution strategy that is most appropriate for a given complex system. The system in question has 5 subsystems, once decomposed, which are fully coupled (Figure 6.10). Each subsystem has one output behavior variable (**Y**). In the analysis, the design variables (**X**) are subsystem-dependent. This coupled system also has an associated optimization, characterized by a system-level objective function, 5 subspaces, and 2 inequality constraints per subspace (Figure 6.11). In the optimization, the design variables are subspace-independent. Simulated evaluation costs have been defined for each behavior variable (random, between 500 and 999), the objective function (a fixed cost of 250), and each constraint function (random, between 0 and 100). The exact values for each function are shown in Table 6.1.

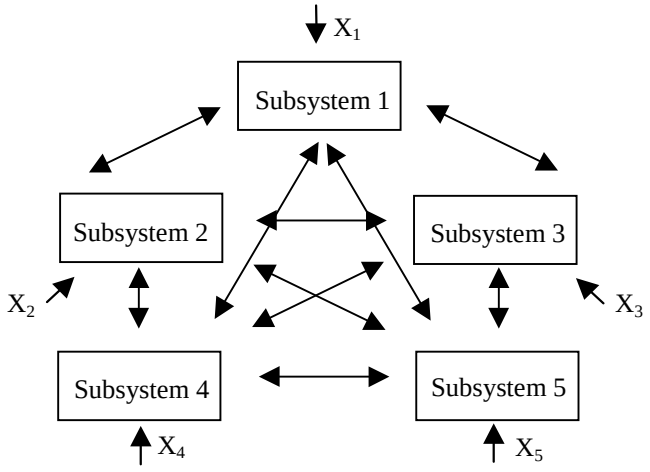


Figure 6.10: Demonstration coupled system

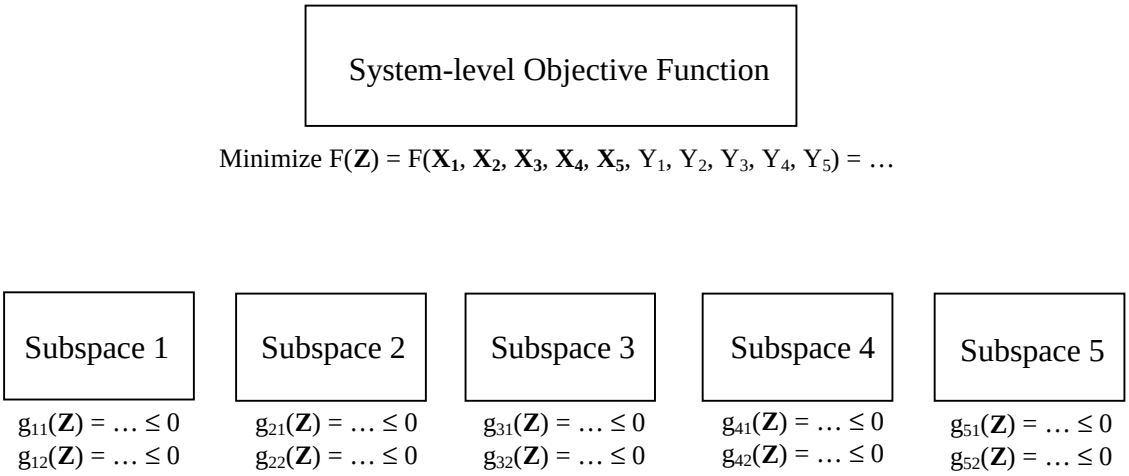


Figure 6.11: Demonstration optimization problem

Table 6.1: Simulated cost values

Function:	**Evaluation Cost:**
Y_1	582
Y_2	514
Y_3	731
Y_4	685
Y_5	510
F	250
g_{11}	3
g_{12}	53
g_{21}	68
g_{22}	41
g_{31}	20
g_{32}	48
g_{41}	92
g_{42}	84
g_{51}	72
g_{52}	44

The user is presented with the “Title window” module upon entering FACETS, as seen in Figure 6.2. After proceeding to the “Directory structure” module (Figure 6.3) and assigning the path structure for file I/O, the user next proceeds to the “Problem

Definition" module (Figure 6.4). Here, the user assigns the desired structure of the simulation of the true system; the semantics are randomly generated. Once the simulated system is generated, the user proceeds to the "Planning" module (Figure 6.5), where basic information about the initial design point is surmised. In this case, the initial objective function has a value of –447.325, and the initial design is feasible. The user next proceeds to the "Optimizer settings" module (Figure 6.6), where optimization settings are assigned in relation to the ADS optimizer. For the present execution, these settings are assigned to a set of "trusted" values and not changed thereafter, for sake of simplicity. The highlights of these settings are seen in Table 6.2.

Table 6.2: Optimization setting highlights

Solution characteristic:	Setting:
Optimization strategy	Sequential Linear Programming
Optimization method	Method of Feasible Directions
Optimization 1-D search	Golden Section Method
Variable scaling?	No
Relative move limit factor	0.75
Equality constraints – Penalty multiplier	2.5
Minimum finite difference change (absolute)	0.1
Required convergence iterations (strategy level)	12
Constraint push-off factor	1.75

After assigning optimization options, the user is finally able to conduct some meaningful research, and proceeds to the "Solution strategies" module (Figure 6.7). Here, the user attempts to solve the MDO problem from the same starting point by way of each of the three primary solution strategies, MDF, IDF, and AAO. The baseline results are seen in Table 6.3. The user can note by viewing the color-coded (shown in gray-scale, here), "Post processing" optimization feedback sub-module that the lowest value of the objective function is attained by the AAO strategy, but that the design point

associated with this value is not feasible (Figure 6.9a). Further, one can see the high cost associated with the MDF strategy, which is the price paid for attaining the largest (feasible) decrease in the objective function (Figure 6.9b). The plotting sub-module (Figure 6.9c) of the post-processor provides objective function vs. Iteration plots, care of XMGR, for each of the three solution strategies. These plots are seen in Figure 6.12. Notice that each solution strategy completes with a different final objective function value, and the paths taken to reach the respective optima are quite distinct.

Table 6.3: Baseline solution strategy results

	MDF	**IDF**	**AAO**
Initial Objective Function	-447.325	-447.325	-447.325
Final objective function	-1025.134	-836.431	-1260.01
Active inequality constraints	4	1	2
Active side constraints	2	2	3
Active equality constraints	n/a	5	2
Violated constraints	0	0	3
Analysis cost	5128334	924732	0
Objective function cost	58000	76250	68250
Constraint function cost	121800	251930	968331
Total cost	5308134	1252912	1036581

At this point, the user has generated a system and has attained a set of baseline results. Typically, the user will view these results, make some intelligent adjustments based on these results, and then attain a new (and hopefully improved) set of results. For instance, the user might wish to verify the validity of the results attained by returning to the “Planning” module to generate a new feasible starting design. In this example, a new feasible design has been generated, and the associated objective function value is 194.334, which is larger than was the initial value. Thereafter, each of the three solution strategies is implemented once again. In this particular setting, the MDF strategy attained

nearly the same solution, IDF attained an infeasible solution, and AAO attained the lowest improvement, but with a feasible final design. This information is summarized in Table 6.4.

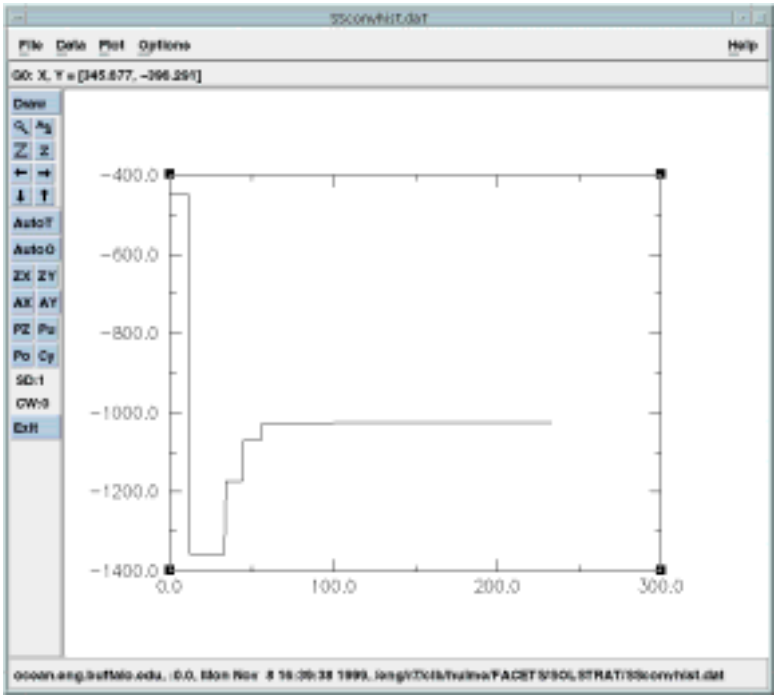
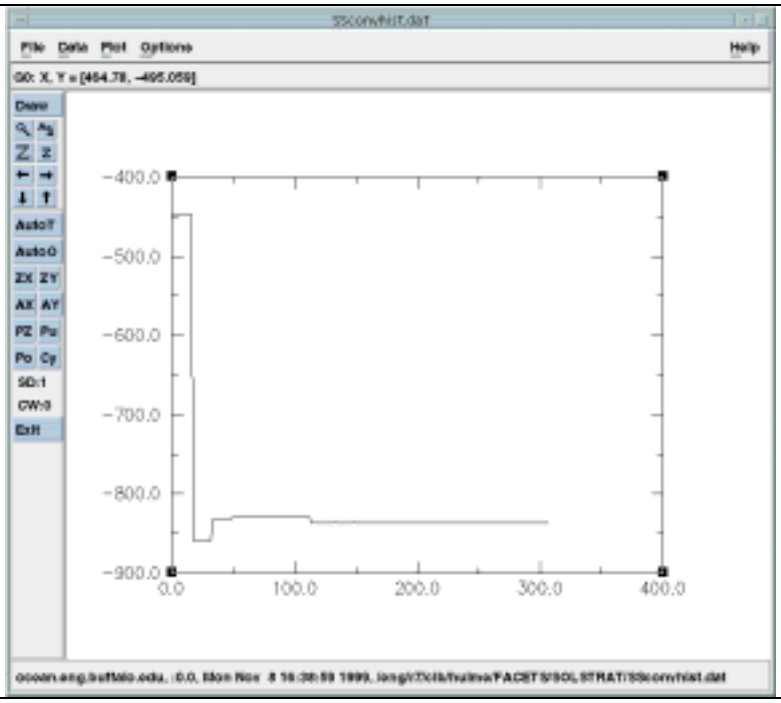
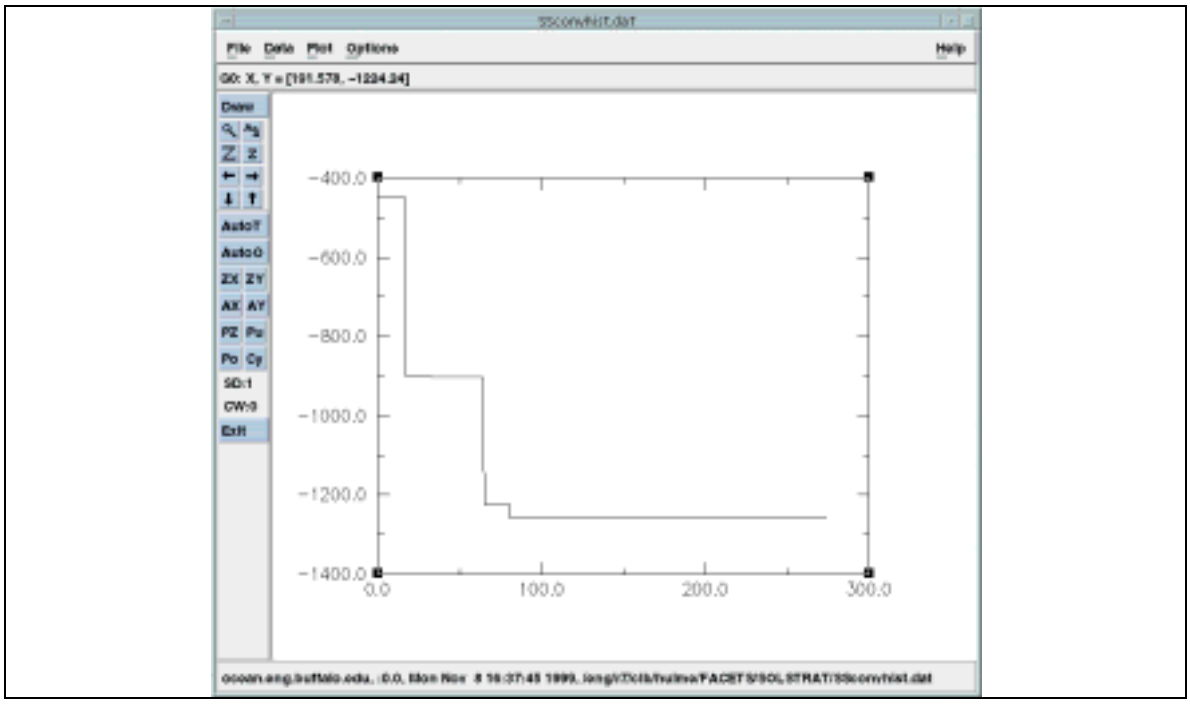


Figure 6.12: Objective function vs. Iteration plots: a. MDF



b. IDF



c. AAO

Table 6.4: Secondary result summary

	MDF	IDF	AAO
Initial Objective Function	194.334	194.334	194.334
Final objective function	-1047.413	-478.366	-136.571
Active inequality constraints	1	2	1
Active side constraints	2	0	0
Active equality constraints	n/a	2	5
Violated constraints	0	3	0

This demonstration usage of FACETS has exemplified how the FACETS framework can be useful to an MDO researcher during the preliminary stages of a multidisciplinary design, in a general sense. Namely, FACETS allows the user to quickly simulate the structure of a real-life coupled system, view its initial characteristics, perform some meaningful baseline calculations in simulation, and then view the initial results. Thereafter, the user can make judgements and subsequent modifications based on these results, and can quickly and easily re-run a new simulation in hopes of attaining

"better" results and more useful insight to the true problem. The heuristic observations made through executing such preliminary simulations may then be put to use during the later phases of the design process. Specifically, these solution heuristics can be implemented when the multidisciplinary design cycle is performed while using the actual system data, as opposed to the simulated design data that is used within FACETS.

The seventh and final chapter of this dissertation presents a multitude of concluding observations, and presents numerous possible avenues for related future research work.

Chapter 7

Conclusions and Future Work

Conclusions

This dissertation has presented an in-depth look at several research developments within the emerging field of Multidisciplinary Design Optimization. While the general procedure of system decomposition - breaking down a large engineering system into a coupled grouping of subsystems - seems like an intuitive approach for engineering design, it has numerous inherent weaknesses. Namely, the structure of the design process that results, though intelligently partitioned, is still a lengthy and costly one. Because of this, many MDO researchers spend a great deal of time designing and developing new strategies for reducing and/or simplifying the MDO process in some way. Many of these techniques were discussed formally in the Background discussion presented in Chapter 2. The primary motivation for this research effort is to reduce the time and cost associated with the multidisciplinary design cycle in any of a number of ways. This research goal was formally stated in Chapter 1, and re-emphasized at the end of the second chapter, as well.

To address this broad and daunting research goal, the present effort has been divided into four primary areas of contribution, which were discussed in detail in Chapters 3-6, respectively. First, the continual design and development of a computer tool called CASCADE, which is used to generate an analytical simulation of a multidisciplinary system. Second, the utilization of CASCADE for the large-scale comparison of numerous popular approaches in MDO for posing and subsequently solving multidisciplinary design problems. Such a comparison had not before been conducted, or even possible. Third, the utilization of both CASCADE and the MDO Test Suite for the comparison of formal and a newly developed heuristic approach for the convergence of a multidisciplinary analysis. The latter approach is entitled Data Fusion Analysis Convergence (DFAC), and is outlined extensively in this work. Fourth and finally, the development of the computational infrastructure that bonds all of these (and other) MDO concepts together: the FACETS framework. Both a summary and conclusions from each of these individual areas of research contribution are now presented.

a. Multidisciplinary System Simulation

Before MDO-based research strategies can be implemented in the design process of such large engineering systems as automobiles and aircraft, they must be tested. A method for analytically simulating real-life, coupled, large engineering systems is necessary. To this end, Chapter 3 presented a discussion of the development of the CASCADE simulator. CASCADE accepts user inputs to randomly construct and then converge a large system of coupled equations. This system of equations can be viewed as an analytical representation to a real-life design scenario. The system has known

structural characteristics, and randomly generated semantic characteristics. The simulator first generates the "system analysis" portion of the design, and then implements Fixed-point Iteration to identify a converged analysis solution. The simulator then builds the optimization problem around this converged design in such a way that the starting design point is feasible. Additional features are continuously being added to the simulator to add to its realism. These include the "system volatility factor", which allows the user to vary the stability of the equations generated. Another new feature is the cost sub-module, which allows the user to assign artificial costs associated with the evaluation of all functions generated by CASCADE.

Once the system is constructed and converged, CASCADE offers numerous post-convergence features. Each of these features is written to an output data or language file, and can be used by other simulation modules downstream. For example, an initial set of total derivatives are computed via the GSE and written to an output file. All of the numerics concerning the final solution point, (namely, the converged values of the design and behavior variables, and the initial values of all optimization functions), are written to a data file. In addition, all equations (analysis behavior variables, objective function(s), and inequality constraints) are written to output files by way of character strings. The output files themselves are ANSI C header files, and can be compiled and linked with other codes such that meaningful research can be conducted.

b. Multidisciplinary Solution Strategies

Chapter 4 presented a numerical comparison of the MDF, IDF, and AAO solution strategies across simulated multidisciplinary design systems of varying size and complexity. The CASCADE simulation tool generated the multidisciplinary test systems.

From the results, some general conclusions can be drawn. Given the ability to pose a multidisciplinary design problem three different ways, a design manager has three distinct means for solving the design problem. This should be viewed as an advantage. Each of the three solution strategies exemplifies a unique problem dynamic, greatly dependent on the initial design point, and in most cases, on the settings of the optimizer.

Numerous figures of merit must be considered when choosing an appropriate solution strategy; the primary of which is design improvement. The MDF strategy has been found to arrive at the "best found" solution for a vast majority of the test problems in this effort. AAO usually attains final solutions that are vastly improved over the initial design without any form of costly iterative system analysis, but at the expense of a complex design space that is cluttered with non-linear equality constraints. IDF has shown itself to be a good trade-off between the extremes of MDF and AAO; fewer analysis evaluations than MDF, and (equilibrium) equality constraints that are simpler, and usually fewer in quantity than those of AAO, depending on the coupling nature of the system. The IDF strategy was typically found to outperform the AAO strategy in terms of objective function improvement.

The reader must realize that the results that have been presented represent MDO problem *simulations*, and must be interpreted as such. In a real-world MDO problem, total solution times for the MDF strategy would likely have been the longest by a considerable margin (due to the many costly iterative analysis evaluations that it requires), followed by IDF (fewer, non-iterative analysis evaluations) and AAO (no iterative analysis evaluations). The cost-based testing supports this line of speculation. MDF was found to be the most costly solution strategy in all three cost-based test

systems, usually by a factor of ten or more. The cost comparison between IDF and AAO is not straightforward. AAO was found to be twice as costly as IDF for two of the three cost-based systems tested, and outperformed IDF (in terms of objective function improvement) in one of these two systems. AAO was found to be slightly less costly than IDF in one cost-based test system, but was slightly outperformed by IDF in this test system.

Finally, it can be reasoned that an improved local (or hopefully global) optimum solution can be found, regardless of the solution strategy used. The ultimate goal of the design manager is to utilize the solution strategy that will achieve the "best found" design with the least difficulty. To this end, the MDF strategy was found to constantly achieve its "best found" solution with minimal modification of optimizer settings, and with the smallest number of trial executions.

c. Multidisciplinary Analysis Convergence

Chapter 5 demonstrated the mathematical details behind a brand new convergence technique for nonlinear, multidisciplinary analysis. This DFAC algorithm is based upon the principals of both Data Fusion and Neural Network theory. The chapter further showed that the DFAC convergence technique was found to be extremely useful for reducing large equation residuals, but not efficient at attaining tight convergence on its own accord. FPI was implemented to supplement the proposed algorithm after a "sufficient" number of iterations had elapsed. This "hybrid" convergence scheme has shown to increase efficiency over the use of conventional FPI by a substantial percentage. Note that due to the comparatively large computational expense of the DFAC convergence technique, it will only be useful in "true" multidisciplinary designs, where

the overall design time is known to be substantially long. In such situations, an increase in computational overhead each iteration will turn out to be beneficial, as it is hoped that the overall number of iterations will be reduced by a sufficient factor, outweighing the added expense.

The general trend seen in the experimental testing is that for "accurate" starting points that turn out to be "close enough" to the true solution, a formal derivative based convergence scheme such as NM is clearly the convergence strategy of choice. NM is reliable in cases where the analysis solution space is known to be convex, which will rarely be the case in most realistic MDO applications. Alternatively, when the starting point is known to be a "sufficiently far" from the converged solution, or if a relation between the initial and final solutions cannot be identified, the DFAC algorithm has shown to be reliable for reducing large residuals. Thereafter, it can be *supplemented* with numerous iterations of FPI (or conceivably NM) to achieve full numerical convergence, thus forming a "hybrid" convergence strategy.

d. MDO Framework Design and Development: FACETS

Chapter 6 presented a discussion of FACETS, a new multidisciplinary design optimization framework that is presently under development. FACETS differs from other MDO frameworks in numerous ways. Namely, it seeks to exploit new tools and technologies in MDO which deal with large-scale, coupled engineering design problems. Because of this, FACETS uses simulation as its backbone. More specifically, the CASCADE simulator has been implemented as its means for generating benchmark problem data upon which new MDO methodologies can be tested. The structure of FACETS has been coded in Motif, an X-Windows based toolkit for GUI programming.

In addition to the problem generation simulator (CASCADE), numerous feature modules have been incorporated into FACETS thus far. These include a module for comparing MDO solution strategies, a module for comparing analysis convergence methods, as well as secondary modules such as planning, optimization, and post processing. Globally speaking, FACETS strives to encompass numerous areas of research in the MDO community, which can provide an "all-in-one" infrastructure for preliminary investigation of MDO solution methods. FACETS has already been extensively used for numerous MDO method comparisons, one of which was presented at the end of the preceding chapter. The benefit of FACETS is that it allows the user to quickly simulate the structure of a real-life coupled system, view its initial characteristics, perform some meaningful baseline calculations in simulation, and then view the initial results. Thereafter, the user can then make judgements and subsequent modifications based on these results, and can quickly and easily re-run a new simulation in hopes of attaining "better" results and more useful insight to the true problem.

Future Work

Improvements and enhancements to the CASCADE simulator are constantly being made. The next phase of its development should attempt to enhance the realism of the equation semantics. Currently, the equations generated are relatively simple, however extremely stable polynomial equations. The addition of logarithmic and trigonometric terms might add to the realism of the simulator, at the possible expense of its stability. The concept initialized by Balling and Wilkinson [4] of generating random equations around a "well-known" structure might be worth expanding upon. In addition to using classical equations from structural optimization, one could conceivably create random

equations using the structure of seminal equations from other engineering disciplines (electrical, aerodynamics, heat transfer, control systems, etc.) as well. Another possibility is the creation of randomly generated differential equations instead of zeroth order polynomial equations. Such equations might represent the semantic behavior of real systems more realistically, and would be sufficiently challenging to solve by either numerical or analytical means.

Minor improvements could be made to the solution strategies module. Three of the primary solution strategies are presently included in the module, along with the "random search" feature that provides a baseline. Numerous variations of these approaches exist, and could be incorporated as explicit solution possibilities. More recently developed large-scale solution strategies such as BLISS [79] and Collaborative Optimization [50,10] could also be incorporated.

The analysis convergence module could also be improved in numerous ways. At present, there are two means in place for conducting a formal iterative convergence procedure: FPI and NM. Countless other algorithms exist, and could be added to the module as options. Examples are the Generalized Gauss-Seidel [25] and the Generalized Jacobi [23] procedures, which have been applied to both systems of algebraic equations and to systems of partial differential equations. In addition, the DFAC algorithm is in its early stages of development, and is constantly undergoing changes. The heuristics of the algorithm must be improved upon so as to supply a solution much more quickly and efficiently. Four fusion algorithms have been developed for the present work. The development of new fusion algorithms based on system characteristics other than sensitivities and equation residuals is envisioned.

There are numerous research avenues that must be pursued to further the FACETS framework. First and foremost, exterior technologies shall be incorporated. All of the major features that lie within the framework at present have been developed and coded exclusively by the author. The next logical step is to incorporate related research technologies performed by other members of the University at Buffalo MODEL laboratory team. These include graph morphing – a means for visualizing and "steering" the multidisciplinary design anytime within the design cycle [88,89]. In addition, coupling suspension strategies – means for identifying and responding to couplings between subsystems which are found to be comparatively weak, in the interest of reducing overall design time and cost [17]. Eventually, technologies developed by completely exterior researchers shall also be incorporated.

Secondly, more optimization possibilities should be added to the framework. Because optimization is the backbone of a majority of the research that FACETS is involved with, alternative options to ADS should be provided. These might include heuristic optimization techniques such as simulated annealing [48], or more recent traditional optimization methods found in Vanderplaats' DOT software package [86].

Some degree of WWW technology shall be incorporated within FACETS. Many of the initial (Java-based) aspirations for this framework have not yet come to pass. At the very least, web technologies can be utilized during the result visualization (post processing) phase of the framework. The automated creation of HTML pages as output files can allow for quick and easy global access of result data attained through the usage of the framework.

Finally, a mention is made regarding the immediate plans for the application and expansion of certain aspects of the present research into a parallel-computing context, for which it is inherently well suited. Namely, the DFAC algorithm is to be parallelized through the use of the PVM [21] message-passing paradigm. The sub-optimizations that are required to reduce the behavior variable residuals to a value of zero on each cycle are easily extended to a parallel-computing scheme. This investigation hopes to improve upon the initial PVM analysis convergence investigation conducted in 1996 [38,39], and discussed in Chapter 3, when Fixed-point Iteration was used as the sole means for system convergence. Recall that the results attained for that investigation demonstrated positive characteristics only after the system equations were augmented with artificial sleep times. This necessity was thought to be due mainly to the inefficiency of passing large arrays element-by-element in Fortran. The complexity of the DFAC algorithm is known to be much higher, and hence much better suited for parallel solution. Moreover, the DFAC algorithm is coded in ANSI C, which lends itself much more advantageously to array passing by virtue of pointer usage.

Initial parallel investigations are to be executed on a sub-network of computers connected via Asynchronous Transfer Mode, or ATM [51,57]. ATM is a relatively new networking technology whereby information is carried on an fiber-optical network, as opposed to a conventional digital signal. Theoretically, transfer rates on optical networks have shown to be “several orders or magnitude” faster than those of their digital counterparts. Though ATM has been designed primarily for the transfer of multimedia-related data (i.e. audio/video), it is theorized that this paradigm can be implemented to enhance the transfer rates associated with numerical data as well. Preliminary

investigations [52,12] have proven ATM to be a favorable alternative to the Ethernet network standard, though actual transfer rates have not yet achieved the performance levels that were initially theorized. First phase testing will conduct a parallel implementation of the DFAC algorithm utilizing both PVM and Fore Systems' built-in ATM API (Application Programming Interface) [20] as means for exchanging data. This investigation will take place on a sub-network of computers consisting of at least two platform types: SUN and SGI workstations.

Second-phase testing will make use of the more commonly used MPI (Message Passing Interface) [73] standard for parallel computing. In a recent comparison between the performance of PVM and MPI [22], it was shown that each has its inherent strengths and weaknesses. Ultimately, PVM is better for applications that will be run over a heterogeneous network of computers; it has good interoperability between hosts, which may reside on multiple platforms. Conversely, MPI is expected to be faster within a large multiprocessor. For this very reason, this second phase of testing will return to the Ethernet networking platform, and will make use of a large, parallel cluster of SUN workstations located at the University at Buffalo Center for Computational Research (CCR).

References

[1] Alexandrov, N.M, and Kodiyalam, S., "Initial Results of an MDO Method Evaluation Study." Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, September 1998, pp. 1315-1327.

[2] Alexandrov, N.M., and Lewis, R.M., "Comparative Properties of Collaborative Optimization and Other Approaches to MDO." First ASMO UK/ISSMO Conference on Engineering Design Optimization, Ilkley, West Yorkshire, United Kingdom, July, 1999, pp. 39-46.

[3] Balling, R. J., and Sobieszczanski-Sobieski, J., "Optimization of Coupled Systems: A Critical Overview of Approaches." AIAA Paper 94-4330-CP, September, 1994, pp. 753-773.

[4] Balling, R.J., and Wilkinson, C.A., "Execution of Multidisciplinary Design Optimization Approaches on Common Test Problems." AIAA Journal, Vol. 35, No. 1, January, 1997, pp. 178-186.

[5] Bloebaum, C. L., "An Intelligent Decomposition Approach for Coupled Engineering Systems." Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, September, 1992, pp. 1177-1187.

[6] Bloebaum, C.L., Hajela, P., and Sobieszczanski-Sobieski, J., "Non-Hierarchic System Decomposition in Structural Optimization." Engineering Optimization, Vol. 19, pp. 171-186, 1992.

[7] Bloebaum, C.L., "Coupling Strength-based System Reduction for Complex Engineering Design." Structural Optimization, Vol. 10, pp. 113-121, 1995.

[8] Box, G.E. and Draper, N.R., "A Basis for the Selection of a Response Surface Design." Journal of the American Statistical Association, Vol. 54, pp. 622-654, September, 1959.

[9] Braun, R.D. et al., "Comparison of Two Multidisciplinary Optimization Strategies for Launch-Vehicle Design." Journal of Spacecraft and Rockets, Vol. 32, No. 3, May-June 1995, pp. 404-410.

[10] Braun, R.D., and Kroo, I.M., "Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment." Multidisciplinary Design Optimization: State of the Art, SIAM, N. Alexandrov and M. Y. Hussaini, editors, SIAM, 1996.

[11] Carter, D. E. and B. S. Baker, "Concurrent Engineering: The Product Development Environment for the 1990's." Addison-Wesley Publishing Company, New York, NY, USA, 1991.

[12] Chang, S., Du, D., Hsieh, J., et al., "Enhanced PVM Communications over a High-Speed Local Area Network." IEEE Parallel & Distributed Technology, Fall 1995, pp.20-32.

[13] Cheney, W., and Kincaid, D., "Numerical Mathematics and Computing." Brooks / Cole Publishing Company, Pacific Grove, 1985.

[14] Cramer, E.J. et al., “On Alternative Problem Formulations for Multidisciplinary Design Optimization.” Fourth AIAA /NASA /ISSMO Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, September, 1992, pp. 518-530.

[15] Cramer, E.J. et al., “Problem Formulation for Multidisciplinary Optimization.” SIAM Journal of Optimization, No. 4, pp. 754-776, November, 1994.

[16] English, K., Miller, E., and Bloebaum, C.L., “Total Derivative-based Coupling Suspension for System Reduction in Complex Design.” Sixth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, September, 1996, pp. 284-294.

[17] English, K, and Bloebaum, C.L., “Development of Multi-cycle Coupling Suspension in Complex System Optimization.” Seventh AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, September, 1998, pp. 107-117.

[18] Eschenauer, Hans A. and Weinert, Matthias, “Approximation Concepts for the Decomposition-Based Optimization of Complex Mechanical Structures on Parallel Computers.” The ASME Design Technical Conferences – 19th Design Automation Conference, Vol. 65-2, pp. 337-345, September, 1993.

[19] Flanagan, D. “Java in a Nutshell, Deluxe Edition.” O'Reilly & Associates, Cambridge, Massachusetts, 1997.

[20] Fore Systems, Inc. “ForeRunner SBA-200 ATM SBus Adapter User’s Manual”, 1993.

[21] Geist, A., Beguelin, A., Dongerra, J., Weicheng, J., Mancheck, R., and Sunderam, V., “PVM: Parallel Virtual Machine - A user's guide and tutorial for networked parallel computing.” The MIT Press, Cambridge, Massachusetts, 1994.

[22] Geist, G.A., Kohl, J.A., and Papadopoulos, P.M., “PVM and MPI: a Comparison of Features.” Calculateurs Paralleles Vol. 8 No. 2, June, 1996, pp. 137-150.

[23] Gourlay, A.R., and Watson, G.A., “Computational Methods for Matrix Eigenproblems.” John Wiley & Sons, New York, N.Y., 1973.

[24] Guisset, P., and Tzennetakis, N., “Numerical Methods for Modeling and Optimization of Noise Emission Applications.” American Society of Mechanical Engineers, Noise Control and Acoustics Division (Publication) NCA, 1997, Vol.24, pp. 315-322.

[25] Hackbusch, W., “Iterative Solution of Large Sparse Systems of Equations.” Springer-Verlag, New York, 1994.

[26] Haftka, R.T., “Simultaneous Analysis and Design.” AIAA Journal, Volume 23, Number 7, July, 1985, pp. 1099-1103.

[27] Haftka, R.T., Sobieszczanski-Sobieski, J., and Padula, S.L., “On Options for Interdisciplinary Analysis and Design Optimization.” Structural Optimization, Volume 4, Number 2, June 1992. pp. 65-74.

[28] Hale, M.A., and Craig, J.I, “Preliminary Development of Agent Technologies for a Design Integration Framework.” Fifth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, FL, September, 1994, pp. 413-422.

[29] Hall, David L., and Llinas, James, “An Introduction to Multisensor Data Fusion.” Proceedings of the IEEE, Vol. 85, No. 1, January, 1997, pp. 6-23.

[30] Hajela, P., “Techniques in Optimum Structural Synthesis with Static and Dynamic Constraints.” Ph.D. Dissertation, Stanford University, 1982.

[31] Hartley, J. R., “Concurrent Engineering. Shortening Lead Times, Raising Quality, and Lowering Costs.” Productivity Press, Cambridge, MA, USA, 1992.

[32] Haykin, Simon, “Neural Networks - A Comprehensive Foundation.” Macmillan College Publishing, New York, 1994.

[33] Heading, A.J.R., and Bedworth, M.D., “Data Fusion for Object Classification.” IEEE Conference on Systems, Man and Cybernetics, Charlottesville, VA, October, 1991.

[34] Heller, D., and Ferguson, P.M., “Motif Programming Manual – Volume 6A.” O’Reilly & Associates, Inc, California, 1994.

[35] Hertz, J., Krogh, A., and Palmer, R.G., “Introduction to the Theory of Neural Computation.” Addison-Wesley Publishing Company, New York, N.Y., 1991.

[36] Huddleston, John V., “Introduction to Computers, FORTRAN version.” Exchange Publishing Division, Buffalo, NY, 1988.

[37] Hulme, K.F., “Development of CASCADE - A Multidisciplinary Design Optimization Test Simulator for Use in Distributed Computing Environments.” Masters Thesis, State University of New York at Buffalo, February, 1996.

[38] Hulme, K.F., and Bloebaum, C.L., "Development of CASCADE - A Multidisciplinary Design Test Simulator." Sixth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, September, 1996, pp. 438-447.

[39] Hulme, K.F., and Bloebaum, C.L., "Development of a Multidisciplinary Design Optimization Test Simulator." Structural Optimization, Volume 14, Number 2-3, October, 1997, pp. 129-137.

[40] Hulme, K.F., and Bloebaum, C.L., "A Comparison of Solution Strategies for Simulation-based Multidisciplinary Design Optimization." Seventh AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, September, 1998, pp. 2143-2153.

[41] Hulme, K.F., and Bloebaum, C.L., "A Comparison of Formal and Heuristic Strategies for Iterative Convergence of a Coupled Multidisciplinary Analysis." Third World Congress on Structural and Multidisciplinary Optimization, Amherst, NY, May, 1999 (CD Proceedings).

[42] Hulme, K.F., and Bloebaum, C.L., "Development of a Simulation-based Framework for Exploiting New Tools and Techniques in Multidisciplinary Design Optimization." First ASMO UK/ISSMO Conference on Engineering Design Optimization, Ilkley, West Yorkshire, United Kingdom, July, 1999, pp. 179-186.

[43] Hulme, K.F., and Bloebaum, C.L., "A Simulation-based Comparison of Multidisciplinary Design Optimization Solution Strategies using CASCADE." Publication pending in "Structural Optimization", 2000 (submitted January, 1999).

[44] Hulme, K.F., and Bloebaum, C.L., "A Data Fusion-based Approach for Coupled Multidisciplinary Analysis." Publication pending in "Design Optimization", 2000 (submitted June, 1999).

[45] Istratescu, V.I., "Fixed Point Theory. An Introduction." Math. and Its Applications, D. Reidel Publishing Co., Dordrecht, 1981.

[46] Kelley, C.T., "Iterative Methods for Linear and Nonlinear Equations." Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[47] Kernighan, B.W., and Ritchie, D.M., "The ANSI C Programming Language, 2nd Edition." Prentice Hall, Englewood Cliffs, N.J., 1988.

[48] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by Simulated Annealing." Science, Vol. 220, pp. 671-680, 1983.

[49] Korsmeyer, D.J., and Walton, J.D., "DARWIN v2 - A Distributed Analytical System for Aeronautical Tests." 20th AIAA Advanced Measurement and Ground Testing Technology Conference, Albuquerque NM, June 1998. AIAA Paper 98-2724.

[50] Kroo, I., Altus, S., Braun, R., Gage, P., and Sobieski, I., "Multidisciplinary Optimization Methods for Aircraft Preliminary Design." Fifth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, FLA, September, 1994, pp. 697-707.

[51] Lakshminarayan, Krishnan, “ATM Networking and Multimedia - A White Paper.” Sun Microsystems Computer Corporation, Revision X, August, 1993.

[52] Lin, M., and Hsieh, J., et al., “Distributed Network Computing over Local ATM Networks.” IEEE Journal on Selected Areas in Communications, Special Issue of ATM LANs: Implementations and Experiences with a Emerging Technology, Vol. 13, No. 4, May 1995, pp.733-748.

[53] Markin, M.S., Harris, C., Bernhardt, M. Austin, J., et al., Data Fusion and Data Processing - The Report of the Defense and Aerospace Foresight Working Party, ISBN 1 85768 0685, May 1997.

[54] McCulley, C., Bloebaum, C. L., “A Genetic Tool for Optimal Design Sequencing in Complex Engineering Systems.” Structural Optimization, Volume 12, Number 2/3, October 1996, pp. 186-201.

[55] McCulley, C., Hulme, K.F., and Bloebaum, C.L., “Simulation-Based Development of Heuristic Strategies for Complex System Convergence.” Applied Mechanics Review, Volume 50, Number 11, Part 2, 1997, pp. S117-S124.

[56] McCulley, C., “Simulation-based Comparison and Development of Heuristic Convergence Strategies for Multidisciplinary Analysis.” Ph.D. Dissertation, State University of New York at Buffalo, 1998.

[57] McDysan, D.E. and Spohn, D.L., “ATM Theory and Application.” McGraw Hill, Inc., New York, N.Y., 1994.

[58] Montgomery, D. C., “Design and Analysis of Experiments, 3rd. ed.” John Wiley and Sons, New York NY, 1981.

[59] Müller, B., Reinhardt, J. and Strickland, M., "Neural Networks -- An Introduction, 2nd ed." Springer Verlag, Heidelberg, 1995.

[60] Musciano, C., Kennedy, B., and Loukides, M., "HTML: The Definitive Guide." O'Reilly & Associates, Cambridge, Massachusetts, 1998.

[61] Padula, S.L., and Young, K.C., "Simulator for Multilevel Optimization Research." NASA Technical Memorandum, Hampton, VA, June, 1986.

[62] Padula, S.L., and Sobieszczanski-Sobieski, J., "A Computer Simulator for Development of Engineering System Design Methodologies." NASA Technical Memorandum, Hampton, VA, February, 1987.

[63] Padula, S.L., Alexandrov, N., and Green, L.L., "MDO Test Suite at NASA Langley Research Center." Sixth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, September, 1996, pp. 410-420.

[64] Peak, D. and Frame, M., "Chaos Under Control." W.H. Freeman and Co., New York, pp. 285-295, 1994.

[65] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., "LU Decomposition and Its Applications." §2.3 in "Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed." Cambridge University Press, Cambridge, England, pp. 34-42, 1992.

[66] Ridlon, S.A., "A Software Framework for Enabling Multidisciplinary Analysis and Optimization." Sixth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, September, 1996, pp. 1280-1285.

[67] Rogers, J.L., “DeMAID -- A Design Manager's Aide for Intelligent Decomposition: User's Guide.” NASA Technical Memorandum 101575, March, 1989.

[68] Rogers, J.L., “DeMAID/GA – An Enhanced Design Manager’s Aid for Intelligent Decomposition.” Sixth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, WA, September, 1996, pp. 1497-1504.

[69] Rogers, J.L., Salas, A.O., and Weston, R.P., “A Web-Based Monitoring System for Multidisciplinary Design Projects.” Seventh AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, September, 1998, pp. 35-43.

[70] Rumelhart, D., Hinton, G. and Williams, R., “Learning Internal Representations by Error Propagation.” Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vols. I and II, pp. 318-362, MIT Press, Cambridge, Massachusetts, 1986.

[71] Salas, A.O., and Townsend, J.C., “Framework Requirements for MDO Application Development.” Seventh AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, September, 1998, pp. 261-271.

[72] Smith, G.D., “Numerical Solution of Partial Differential Equations: Finite Difference Methods.” Oxford University Press, 1985.

[73] Snir, M., Otto, S., et al., “MPI: The Complete Reference (Vol. 1) - 2nd Edition. Volume 1 - The MPI Core.” The MIT Press, Cambridge, Massachusetts, 1998.

[74] Sobieszczanski-Sobieski, J., "A Linear Decomposition Method for Optimization Problems - Blueprint for Development." NASA Technical Memorandum 83248, 1982.

[75] Sobieszczanski-Sobieski, J., "Optimization by Decomposition: A Step from Hierarchic to Non-Hierarchic Systems." Recent Advances in Multidisciplinary Analysis and Optimization, NASA CP 3031, 1988.

[76] Sobieszczanski-Sobieski, J., "The Sensitivity of Complex, Internally Coupled Systems." AIAA Journal, Volume 28, No. 1, 1990, pp. 153-160.

[77] Sobieski, J., Bloebaum, C. L., and Hajela, P., "Sensitivity of Control-Augmented Structure Obtained by a System Decomposition Method." AIAA Journal, Volume 29, Number 2, 1991, pp. 264-270.

[78] Sobieszczanski-Sobieski, J., "Multidisciplinary Design Optimization: An Emerging, New Engineering Discipline." Advances in Structural Optimization, pp. 483-496, Kluwer Academic, 1995.

[79] Sobieszczanski-Sobieski, J., Agte, J.S., and Sandusky Jr., R.R., "Bi-Level Integrated System Synthesis (BLISS)." Seventh AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, September, 1998, pp. 1543-1557.

[80] Steward, D.V., "System Analysis and Management". Petrocelli Books, New York, N.Y., 1981.

[81] Swan, T., "Mastering Borland C++." Sams Publishing, Indianapolis, IN, 1992.

[82] Tong, S.S., Powell, D., and Goel, S., “Integration of Artificial Intelligence and Numerical Optimization Techniques for the Design of Complex Aerospace Systems.” AIAA Paper 92-1189, February, 1992.

[83] Turner, P.J. “XMGR – XY plotting tool for workstations using X”, Internet Documentation. Portland, OR, 1991-1995.

[84] Vanderplaats, G. N., “Numerical Optimization Techniques for Engineering Design: with Applications.” McGraw Hill, New York, N.Y., 1984.

[85] Vanderplaats, G., “ADS - A FORTRAN Program for Automated Design Synthesis Version 1.10”, User's Manual. University of California, Santa Barbara, California, 1985.

[86] Vanderplaats, G., “Design Optimization Tools (DOT) – User’s Manual v5.0.” Vanderplaats Research and Development, Colorado Springs, 1999.

[87] Weston, R.P., Townsend, J.C., et al, “A Distributed Computing Environment for Multidisciplinary Design.” Fifth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, FL, September, 1994, pp. 1091-1097.

[88] Winer, E., and Bloebaum, C.L., “Design Visualization by Graph Morphing for Multidisciplinary Design Optimization”, Conference Proceedings of First International Conference on Engineering Design and Automation (EDA '97), Bangkok, Thailand, 1997 (CD Proceedings).

[89] Winer, E., and Bloebaum, C.L., “N-Dimensional Design Visualization via Graph Morphing for Large-Scale Optimization.” Second World Congress of Structural and Multidisciplinary Optimization, Zakopane, Poland, 1997, pp. 911-916.

[90] Zangwill, Willard I., “Nonlinear Programming: A Unified Approach.” Prentice-Hall, Englewood Cliffs, N.J., 1969.

[91] Zhigljavsky, A.A., “Theory of Global Random Search.” Kluwer Academic Publishers, Dordrecht / Boston / London, 1991.

Appendix I - DFAC CASCADE test systems

Test System	Design Variables			Behavior variable equations	Initial guess		
	X0	X1	X2		Y0	Y1	Y2
CASCADE #1	8132.73	8510.5	5275.96	$y_0 = -2.167e\text{-}09x_0^3 + 6.013e\text{-}01y_1^1 + 5.975e\text{-}01y_2^1$	9664.56	3542.43	3978.78
				$y_1 = +4.012e\text{-}09x_1^3 + 2.773e\text{-}09y_0^3 + 4.164e\text{-}01y_2^1$			
				$y_2 = +2.609e\text{-}05x_2^2 + 6.021e\text{-}01y_0^1 + 1.460e\text{-}02y_1^1$			
CASCADE #2	250.0	325.0	87.5	$y_0 = x_0 + 0.25y_1 + 0.0015y_2^2$	-822.0	47.0	629.0
				$y_1 = x_1 - 7.5e\text{-}7y_0^3 - 0.095y_2$			
				$y_2 = -x_2 + 0.00095y_0^2 - 3.8e\text{-}06y_1^3$			
CASCADE #3	6873.39	11.29	4506.99	$y_0 = +7.695e\text{-}01x_0^1 - 6.779e\text{-}01y_1^1 + 9.191e\text{-}02y_2^1$	8830.29	1194.64	6283.55
				$y_1 = +9.176e\text{-}09x_1^3 + 1.260e\text{-}04y_0^2 + 7.707e\text{-}05y_2^2$			
				$y_2 = +7.285e\text{-}05x_2^2 - 1.124e\text{-}04y_0^2 + 3.124e\text{-}09y_1^3$			

Appendix II – DFAC MDO Test Suite test systems

	Design Variables	Constants	Behavior variable equations	Initial guess
Heart Dipole $k_i = 1.0$, i=1,8 $q_0 = x_4^2 - x_6^2$ $q_1 = x_5^2 - 3x_7^2$	$x_0 = 0.4285$	$c_0 = 0.2372$	$y_0 = c_0(k_0 - y_1)$	$y_0 = 0.567$
	$x_1 = 0.3203$	$c_1 = 0.5446$	$y_1 = c_1((k_1 + (y_3y_7(y_7^2 - 3y_5^2)) - (y_2y_6(y_6^2 - 3y_4^2)) - (y_4^2 - 3y_6^2)) / (x_5q_1))$	$y_1 = 0.144$
	$x_2 = 0.7332$	$c_2 = 0.1754$	$y_2 = c_2(k_2 - y_3)$	$y_2 = 0.591$
	$x_3 = 0.0770$	$c_3 = 0.8407$	$y_3 = c_3((k_3 + (y_1y_7(y_7^2 - 3y_5^2)) + (y_0y_6(y_6^2 - 3y_4^2)) - (y_2y_4(y_4^2 - 3y_6^2))) / (x_5q_1))$	$y_3 = 0.144$
	$x_4 = 0.3376$	$c_4 = 0.2124$	$y_4 = c_4((k_4 + (2y_3y_5y_7) - (y_1(y_5^2 - y_7^2)) - (y_0q_0)) / (-2x_2x_6))$	$y_4 = 0.529$
	$x_5 = 0.7378$	$c_5 = 0.0822$	$y_5 = c_5((k_5 + (y_7y_3) + (y_6y_2) - (y_4y_0)) / x_1)$	$y_5 = 0.045$
	$x_6 = 0.5415$	$c_6 = 0.0203$	$y_6 = c_6((k_6 - (2y_1y_5y_7) - (y_3(y_5^2 - y_7^2)) - (y_2q_0)) / (2x_0x_4))$	$y_6 = 0.499$
	$x_7 = 0.0737$	$c_7 = 0.2626$	$y_7 = c_7((k_7 - (y_5y_3) - (y_4y_2) - (y_6y_0)) / x_1)$	$y_7 = 0.506$
Combustion of Propane P = 40.0 R = 10.0		$k_0 = 0.2345$	$y_0 = k_0(- y_3 + 3)$	$y_0 = 0.055$
		$k_1 = 0.4527$	$y_1 = k_1(- y_4 - (y_5/2) - (y_6/2) + 4)$	$y_1 = 0.980$
		$k_2 = 0.5174$	$y_2 = k_2(- (y_8/2) + (2R))$	$y_2 = 0.728$
		$k_3 = 0.1247$	$y_3 = k_3(-(2y_0) - y_1 - y_6 - y_7 - y_8 - (2y_9) + R)$	$y_3 = 0.153$
		$k_4 = 1.0$	$y_4 = (k_4y_1y_3)/y_0$	$y_4 = 0.412$
		$k_5 = 1.0$	$y_5 = (k_5y_1^{0.5}y_3^{0.5}) / (y_0^{0.5}(P / y_{10})^{0.5})$	$y_5 = 0.145$
		$k_6 = 1.0$	$y_6 = (k_6y_0^{0.5}y_1^{0.5}) / (y_3^{0.5}(P / y_{10})^{0.5})$	$y_6 = 0.907$
		$k_7 = 0.1$	$y_7 = (k_7y_0) / (y_3(P / y_{10}))$	$y_7 = 0.408$
		$k_8 = 1.0$	$y_8 = (k_8y_0y_2^{0.5}) / (y_3(P / y_{10})^{0.5})$	$y_8 = 0.292$
		$k_9 = 0.1$	$y_9 = (k_9y_0^{0.5}) / (y_3^2(P / y_{10}))$	$y_9 = 0.219$
			$y_{10} = y_0 + y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9$	$y_{10} = 0.660$